Univerza v Ljubljani
Fakulteta za računalništvo in informatiko

Andraž Hribar

Mobilne tehnologije pri razvoju osebnega zdravstvenega kartona

diplomsko delo

univerzitetni študijski program prve stopnje
računalništvo in informatika

Mentor: izr. prof. dr. Uroš Lotrič
Somentor: Rok Vrbica

Ljubljana 2015

Besedilo je oblikovano z urejevalnikom besedil LaTeX.

Fakulteta za računalništvo in informatiko izdaja naslednjo nalogo:

Tematika naloge:

Mobilne naprave postajajo vedno bolj nepogrešljiv pripomoček tudi na področju medicine. Imamo jih vedno pri roki, zato lahko z njimi neprestano spremljamo tudi svoje zdravje in življenjski slog. Raziščite stanje mobilnih aplikacij s področja zdravstva in izdelajte prototip mobilnega osebnega zdravstvenega kartona za operacijski sistem Android. Ta naj uporabniku omogoča vnos podatkov in integracijo z informacijskimi sistemi zdravstvenih ustanov. Pri tem se v največji meri držite obstoječih standardov za izmenjevanje podatkov.

Izjava o avtorstvu diplomskega dela

Spodaj podpisani Andraž Hribar, z vpisno številko 63110222, sem avtor diplomskega dela z naslovom:

Mobilne tehnologije pri razvoju osebnega zdravstvenega kartona

S svojim podpisom zagotavljam, da:

- sem diplomsko delo izdelal samostojno pod mentorstvom izr. prof. dr. Uroša Lotriča in somentorstvom Roka Vrbice,
- so elektronska oblika diplomskega dela, naslov (slov., angl.), povzetek (slov., angl.) ter ključne besede (slov., angl.) identični s tiskano obliko diplomskega dela,
- soglašam z javno objavo elektronske oblike diplomskega dela na svetovnem spletu preko univerzitetnega spletnega arhiva.

V Ljubljani, dne 8. septembra 2015 Podpis avtorja:

Zahvaljujem se mentorju izr. prof. dr. Urošu Lotriču za vodenje in pomoč pri diplomski nalogi ter celotni ekipi podjetja Codemonkee za vso pripravljenost, strokovno podporo in sproščeno vzdušje pri izvedbi projekta. Prav tako bi se zahvalil dr. Majdi Ambrož Mihelčič za koristne strokovne nasvete.

Iskreno pa se zahvaljujem tudi staršem za vso podporo in potrpežljivost tekom celotnega študija.

Kazalo

Slike

Seznam uporabljenih kratic

ADL	Archetype Definition Language
API	Application Programming Interface
APK	Android Package
CDA	Clinical Document Architecture
CRUD	Create, Read, Update, Delete
CT	Computed Tomography
DAO	Data Access Object
DICOM	Digital Imaging and Communications in Medicine
EHR	Electronic Health Record
EMR	Electronic Medical Record
EPM	Electronic Practice Management
epSOS	Smart Open Services for European Patients
ER	Entity Relationship
HL7	Health Level 7
HTTP	HyperText Transfer Protocol
IDE	Integrated Development Environment
IHE	Integrating the Healthcare Enterprise
IT	Information Technology
LIS	Laboratory Information System
MR	magnetna resonanca
ORM	Object-Relational Mapping
OS	operacijski sistem
PAS	Patient Administration System

PHR	Personal Health Record
PDA	Personal Digital Assistant
REST	Representational State Transfer
SQL	Structured Query Language
SSL	Secure Sockets Layer
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UZ	ultrazvok
XML	Extensible Markup Language

Povzetek

Sledenje svojim življenjskim navadam postaja vedno bolj priljubljena tematika aplikacij za pametne mobilne naprave, saj omogočajo uporabniku širok pregled nad stanjem svojega zdravja ali načina življenja. V diplomskem delu je opisana realizacija prototipa mobilnega zdravstvenega kartona za operacijski sistem Android, katerega namen je vodenje posameznikove zdravstvene evidence in deljenje kliničnih informacij med konkretnimi zdravstvenimi sistemi. Deljenje in pošiljanje podatkov neodvisno od sistemskega ali programskega okolja pa je pogojeno z uporabo protokolov in standardov. Opisana sta tudi najbolj razširjena standarda za izmenjavo medicinskih informacij, HL7 in openEHR, ter implementacija in uporaba standarda na področju Evropske unije, epSOS.

Ključne besede: medicinska informatika, mZdravje, PHR, epSOS.

Abstract

Tracking one's lifestyle is becoming an increasingly popular topic regarding mobile application development, since it enables the user to have a broad overview of his or her health or lifestyle status. This thesis describes the realization of a prototype personal health record for the Android operating system, which aims at managing the user's medical records and sharing of clinical information between concrete health systems. Sharing and sending independently of the system or programming environment is conditioned by the use of protocols and standards. In addition, the most widespread standards for exchanging medical information, standards HL7 and openEHR, with the implementation and use of a standard in the European Union, epSOS, are also described.

Keywords: medical informatics, mHealth, PHR, epSOS.

Poglavje 1

Uvod

Osebni zdravstveni karton je po definiciji zbirka medicinskih podatkov, ki jih pacient ureja zase. Sodobna tehnologija in predvsem mobilne naprave omogočajo, da so realizacije takega kartona vedno bolj zmogljive, povezljive in tudi prijazne do uporabnika. Čeprav gre pri osebnem zdravstvenem kartonu za zapisovanje določenih medicinskih količin, kar samo po sebi ne bi smelo biti zapleteno, je v praksi drugače. Pri izmenjavi podatkov imamo opraviti z vprašanji standardov, pri shranjevanju podatkov z varovanjem osebnih podatkov, posebna pozornost pa gre tudi uporabniški izkušnji, saj imamo opraviti z vnosi in pregledi kompleksnih podatkov na majhnih zaslonih.

Namen dela Mobilne tehnologije pri razvoju osebnega zdravstvenega kartonaje pregled stanja razvitosti sodelovanja med medicino in informatiko v smislu izmenjave podatkov med informacijskimi sistemi zdravstvenih ustanov ali posameznimi pacienti ter implementacija osebnega zdravstvenega kartona za mobilne naprave sistema Android.

Kako sodelujeta medicina in informatika? Prva ideja je verjetno nekaj v smislu povezave naprav za kompleksne preiskave in zmogljivih računalnikov, ki znajo podatke iz naprave procesirati in prikazati v zdravstvenemu osebju berljivem formatu. Ali pa morda računalniško vodeni procesi zdravstvene nege, kjer nas namesto zdravnika pregleda in vodi umetna inteligenca. V

poglavju 2 spoznamo, da področje medicinske informatike še zdaleč ni tako ozko omejeno samo na okolje znotraj zdravstvenih domov in bolnišnic ter da še zdaleč ni tako razvito, kakor nas prepričujejo znanstveno fantastični filmi. V nadaljevanju se osredotočimo na tisti del medicinske informatike, ki je zadolžen za izmenjavo kliničnih podatkov med zdravstvenimi ustanovami ali pacienti. Poglavje 3 govori o konceptih standardizacije načina komunikacije zdravstvenih informacijskih sistemov. V poglavju 4 pa preidemo iz teoretičnih osnov na praktično uporabo. Tu spoznamo bistvene principe, uporabljene pri izdelavi mobilnega osebnega zdravstvenega kartona, od osnov platforme do višje nivojskih implementacij aplikacije.

Poglavje 2

Medicinska informatika

Kvaliteta človeškega življenja je odraz zadovoljevanja njegovih potreb in ena glavnih je potreba po zdravju. Le-to poskušajo zdravstvene ustanove zagotavljati s kompleksnimi procesi in sistemi. A kot vsak sistem ima tudi ta pomanjkljivosti, ki jih je treba odpraviti. S pojavom računalnika se je odprlo mnogo konceptov, ki bi lahko zdravstveni sistem izboljšali. Področje v računalništvu, ki se ukvarja z obdelavo informacij o medicinskih praksah, izobraževanju in raziskovanju v sodelovanju z informatiko in medicinsko podprtimi tehnologijami, imenujemo medicinska informatika.

2.1 eZdravje

Ko govorimo o medicinski informatiki, nikakor ne moremo mimo pojma eZdravje (ang. eHealth). Ta se je uveljavil po letu 1999 z uvajanjem e-besed, ki so bile plod industrijskega trženja, zaradi česar danes obstaja preko 50 različnih definicij tega pojma, bodisi v smislu poslovne definicije bodisi v smislu akademske. Najbolj splošno lahko pojem eZdravje definiramo kot razvijajoče se področje prepletanja medicinske informatike, javnega zdravstva in poslovanja, v smislu zdravstvenih storitev in posredovanja informacij s pomočjo računalniške tehnologije. Pojem ne vpeljuje samo tehnološkega razvoja, ampak tudi način razmišljanja ter predanost globalnemu povezovanju

in posredovanju informacij za izboljšanje zdravstvene nege [17].

2.2 mZdravje

Kmalu po uveljavitvi računalniškega obdelovanja medicinskih informacij se je pojavila ideja o dostopanju do podatkov od koderkoli, kadarkoli. Platforme za ta namen so že razvite (PDA, pametni mobilni telefoni, tablice ...), potrebna je le še programska oprema, ki zna zdravstvene informacije prebrati, jih obdelati in smiselno prikazati. Iz eZdravja se je razvilo novo, hitro rastoče področje, ki se ukvarja s temi koncepti, imenovano mZdravje (ang. mHealth). Poleg razvijanja aplikacij, ki so strogo povezane z medicino, to področje zajema tudi razvoj aplikacij, ki uporabniku omogočajo sledenje svojemu življenjskemu slogu[20] (tipični primeri aplikacij: za Android Google Fit ali S Health, za iOS Lifesum). Takšne aplikacije lahko uporabniku pomagajo spremeniti način življenja ali mu olajšajo vsakodnevne dejavnosti s tem, ko mu narekujejo urnik prehranjevanja, športnih aktivnosti ali jemanja zdravil, značilna pa je tudi uporaba raznih motivacijskih gest. Z uporabo ločenih pripomočkov, na primer brezžičnega merilca srčnega utripa, ali z uporabo integriranih senzorjev v sami napravi, na primer GPS ali merilnik pospeškov, lahko z aplikacijo izmerimo različne življenjske znake, od preprostih meritev korakov do kompleksnejših meritev krvnega tlaka ali sladkorja. Te meritve so zelo dober pokazatelj zdravniku, saj so meritve navadno opravljene v sproščenem okolju namesto v bolnici ali zdravstvenemu domu, ki bi lahko na subjekt deloval stresno. Glavni namen je samostojno vodenje in nadziranje svojega načina življenja in ne nadomestitev usposobljenih zdravniških delavcev. Zanje so ti podatki le dodatni vir informacij iz drugega zornega kota.

2.2.1 Pregled stanja po svetu

Razvoj aplikacij za področje mZdravje je odvisen od dveh kriterijev, to sta pripravljenost oziroma dovzetnost okolice do uporabe tovrstnih aplikacij ter

usposobljenost in interes razvijalcev [14]. Pred raziskavo trga je treba preveriti državne zakone in se prepričati, če je razvijanje takšnih aplikacij sploh legalno s stališča varovanja osebnih podatkov. Drugi pomemben podatek je pripravljenost sodelovanja zdravstvenih ustanov. Če ustanove niso pripravljene ali ne morejo zagotoviti sredstev za investicijo, za razširitev svoje informacijske tehnologije, je razvoj aplikacij, ki temeljijo na komunikaciji med zdravniki in pacienti, nesmiseln. Tretja omejitev potencialnega trga je nivo ozaveščenosti potencialnih uporabnikov. Tudi če posredovanje pacientovih aplikacij ni v sporu z nobenim zakonom in če so zdravstvene ustanove pripravljene migrirati ali posodobiti na informacijski sistem, se še vedno pojavlja vprašanje kupcev ali uporabnikov, ki je pri razvoju in prodaji mobilnih aplikacij v katerikoli smeri najbolj bistvenega pomena.

Po drugi strani pa je vprašljiva tudi izvedba samega razvoja. Pojavi se vprašanje pripravljenosti tveganja podjetja, ki bi se podalo v razvoj takšne aplikacije, in usposobljenosti zaposlenih. Načrtovanje aplikacije takšnega tipa je zapleten postopek, saj je treba pregledati in upoštevati že uveljavljene standarde za delo s kliničnimi podatki ter preveriti, ali zdravstvene ustanove morda že uporabljajo kakšen standard, bodisi mednarodno priznanega bodisi svojega. Težava nastane, če pride do situacije, ko vsak informacijski sistem ustanove uporablja svoj standard, razvijalec pa želi ustvariti aplikacijo, ki bi univerzalno znala prejemati podatke vseh ponudnikov. Implementacija posameznih bralnikov je zamudna in s seboj prinese ogromne stroške.

Pri podjetju research2guidance [16] od leta 2010 izvajajo raziskave trga v okviru programa mHealth App Developer Economics za področje mZdravja. Letošnja raziskava je na podlagi dobrih 4400 anketiranih podjetij iz 28 evropskih držav, ki proizvajajo aplikacije na področju mZdravja, pokazala, da se stanje trga razlikuje od države do države. Začetek razvoja je odvisen od trenutnega položaja uporabe rešitev mZdravja, zato je nujno, da se ta položaj oceni z določenimi kazalci. V splošnem je definiranih pet tržnih dimenzij, ki so pomembne za investiranje ali razvoj aplikacije tovrstne smeri [16]. Te so:

- iskanje mednarodnih institucij, ki podpirajo razvoj aplikacij na po-

dročju mZdravje,

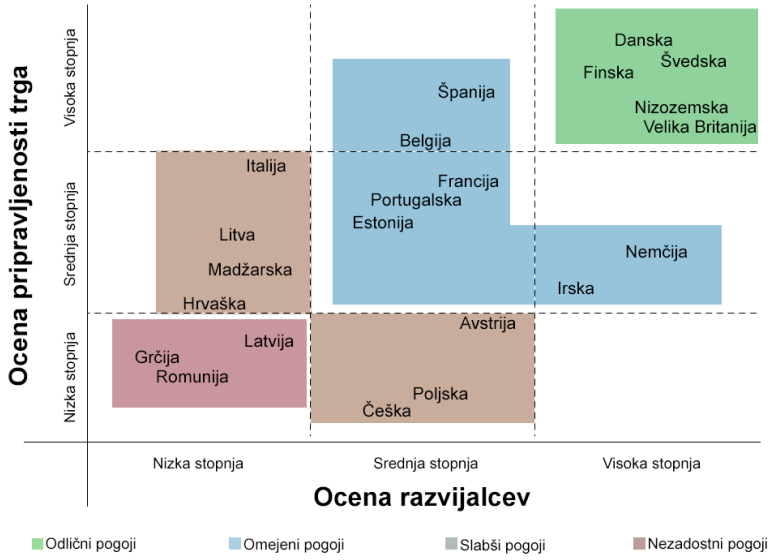
- razumevanje tržnih kriterijev in pogojev, ki vplivajo na poslovni načrt,
- razumevanje delovanja zdravstvenega sistema,
- premišljeno izbiranje ciljne države za trženje aplikacije - najbolj priporočljivih je 5 držav, z najvišje uvrščenim zanimanjem trga za uporabo rešitev mZdravje, prikazano v Shemi 2.1,
- sodelovanje z razvijalci in ponudniki rešitev mZdravja v izbranih državah in
- raziskava trga - test pripravljenosti pacientov in zdravnikov za koriščenje storitev, ki jih nudi rešitev mZdravje.

Promoviranje uporabe rešitev mZdravja pa ni odvisno le od ponudnikov, temveč tudi od državnih zdravstvenih institucij, saj je uporaba proizvodov mZdravja koristna ne samo za uporabnike, ampak tudi za celoten zdravstveni sistem v smislu izboljšanega nudenja zdravniške oskrbe in njenih zmanjšanih stroškov. Raziskava je pokazala, da sta na območju Evropske unije z ukrepi za uveljavljanje storitev mZdravje začeli le Danska in Španija. S tem sta pridobili konkurenčno prednost pred ostalimi državami, kar pripomore k izboljšanju atraktivnosti tržišča za ponudnike storitev [16]. Ukrepi, ki lahko pripeljejo do izboljšanja stanja trga, so:

- razumevanje dejstva, da so podjetja, ki ponujajo rešitve mZdravje, dragoceni partnerji, ki pomagajo modernizirati zdravstveni sistem,
- upoštevanje raziskav trga, ki jih izvajajo posamezna podjetja za lastne potrebe, saj se tu skriva ocena stanja,
- snovanje smernic in ukrepov za izboljšanje zdravstvenega sistema,
- izobraževanje zdravniškega osebja o možnostih, ki jih prinašajo rešitve mZdravja,

- sprejemanje iniciative za digitalizacijo zdravstvenega sistema.

Slika 2.1 prikazuje graf ugodnosti za razvoj aplikacije v posameznih državah in oceno razvijalcev. Navpična os predstavlja stopnjo pripravljenosti trga države glede na razširjenost uporabe rešitev eZdravja, stopnjo digitalizacije med državljani, uspešnost trga, zahtevnosti ustanovitve podjetja in regulacij mZdravja, vodoravna os pa predstavlja stopnjo pripravljenosti razvijalcev. Približno polovica anketiranih držav ima enako stopnjo pripravljenosti trga in razvijalcev, medtem ko imajo razvijalci v Nemčiji in Avstriji ter na Irskem, Poljskem in Češkem prevelika pričakovanja od trga. Države, ki imajo stopnjo pripravljenosti trga višjo od stopnje razvijalcev, kot je na primer Španija, ki ima odlično pripravljen zdravstveni sistem za implementacijo storitev mZdravja, pa še niso uspele prepričati razvijalcev mobilnih aplikacij.



Slika 2.1: Graf raziskave stanja trga in razvoja, povzeto po [16].

2.2.2 Pregled aplikacij mZdravja

Evropska unija se je leta 2014 zavzemala za razvoj projektov na področju mZdravja, saj je bilo v njih v roku dveh let investiranih preko 100 milijonov

evrov. Na uradni strani Evropske komisije je objavljeno poročilo [15] o štirih najbolj izpostavljenih in uspešnih projektih.

Nephron Plus - prvi projekt se nanaša na paciente z nefrološkimi težavami, grške organizacije NEPHRON+ (dosegljivo na `http://www.nephronplus.eu`). Projekt je prejel 5 milijonov evrov evropskih sredstev, gre pa za napravo, ki nadomešča funkcijo ledvic. Napravo pacient vedno nosi s seboj, podobno kot ima sladkorni bolnik avtomatsko črpalko. Delovanje te umetne ledvice lahko pacient spremlja kar preko pametne mobilne naprave. Namen naprave je celodnevno izvajanje prečiščevanja organizma namesto izvajanja dialize enkrat dnevno. Po hemodializi namreč pacient pogosto občuti slabost zaradi prevelikega ali prehitrega odvzema tekočin iz krvnega obtoka, z Nephron Plus pa se kri prečiščuje počasi s konstantnim časom, zato do slabosti ne prihaja. Sprotno pregledovanje stanja ledvice pacientu omogoča prilagajanje dani situaciji.

REACTION - namen projekta REACTION (dosegljivo na `http://www.reaction-project.eu`) je deljenje informacij pacientov med zdravstvenim osebjem. Projekt skuša rešiti težavo pomanjkljivega informiranja osebja zaradi urgentnih primerov, menjave izmen in splošne preobremenjenosti osebja. Projekt REACTION temelji na centraliziranem ponudniku informacij pacientov, do katerega bi zdravstveno osebje lahko dostopalo kar preko mobilnih naprav ali tabličnih računalnikov. Informacije bi bile zajete tako ročno (zdravniški zapisi in navodila) kot avtomatsko preko senzorjev naprav, ki spremljajo pacienta (na primer krvni tlak ali nivo sladkorja v krvi). Ta sistem je že v uporabi v avstrijski zdravstveni univerzi v Gradcu, odziv osebja pa je zelo pozitiven. Projekt je prejel tudi oznako CE, s čimer je omogočeno, da se rešitev implementira v vsa bolnišnična okolja, prejel pa je tudi avstrijsko nagrado za najboljšo komercialno aplikacijo na področju komunikacije človek-računalnik. REACTION omogoča tudi sledenje načinu življenja. Primarno je bila ta funkcionalnost razvita za paciente s sladkorno boleznijo, uporablja pa jo zdravstveni center v Chorleywoodu v Veliki Britaniji, kjer se med zaposlenimi delijo informacije o pacientovem nivoju sladkorja v krvi,

teži in višini ter ostale lastnosti.

MobiGuide - projekt MobiGuide je pametni mobilni sistem za pomoč pacientom s kroničnimi boleznimi, primarno je namenjen bolnikom s težavami na področju bolezni srca in ožilja ter ženskam s poporodnimi težavami, konkretno pa je namenjen za sladkorno bolezen med in po nosečnosti, uporaba sistema bo možna tudi za ostale bolezni. Razvit je bil pod okriljem izraelske univerze v Haifu. Predvidena je uporaba perifernih merilnih naprav, ki meritve pošljejo neposredno na pametno mobilno napravo z naloženo aplikacijo, ki informacije posreduje računalniku s strežniško programsko opremo MOBIGUIDE, ki omogoča opravljanje analiz. Analizo lahko opravijo tudi tako, da upoštevajo subjektove starejše podatke, kar poveča natančnost preiskave. Po potrebi lahko aplikacija od uporabnika zahteva razne odgovore na vprašanja, povezana s preiskavo. Rezultat analize je predlog spremembe načina življenja, ki bi uporabniku najbolj pomagal pri lajšanju težav njegove/ih bolezni. Taisti rezultat je obenem posredovan tudi izbranemu zdravniku pacienta.

Interstress - pod zadnjim projektom, navedenim v poročilu, imenovanim Interstress (Istituto Auxologico Italiano), je bila narejena aplikacija Positive Technology App, ki je na voljo za naprave s sistemom iOS preko spletne trgovine iTunes. Gre za protistresno aplikacijo, kjer se uporabnik sprošča preko vizualizacije umirjenih motivov (ogenj, slap, narava ...). Aplikacija uporabnika nauči tehnike sproščanja, poveže pa se tudi lahko z biosenzorji (aktivne zapestnice), kar omogoči takojšnjo interakcijo med uporabnikom in aplikacijo, saj uporabnik nemudoma prejme obvestilo o povišanju stresa.

2.2.3 Tehnološki potencial

Preventivno iskanje zdravniške pomoči ali mnenja – rešitev v sklopu mZdravja bi lahko avtomatsko z analizo vnesenih podatkov zaznala možno nastajanje bolezni že v zgodnjih fazah in subjektu priporočila obisk pri zdravniku zaradi točno določenega simptoma, nadaljnje izvajanje ukrepov pa je odvisno od zdravniškega mnenja. Tu sicer lahko nastane težava, ko lahko z ne-

premišljenim in prezgodnjim opozorilom povzročimo nepotrebno paniko, zato bi bilo treba med razvijanjem takšnega sistema postaviti dobro premišljene omejitve in vzorce za odkrivanje. Če pogledamo tako rešitev iz ekonomskega vidika, opazimo, da nam lahko tak način odkrivanja bolezni prihrani ogromno denarja. Namesto dragih, zapletenih in ne vedno uspešnih posegov bi izvajali cenejše in manj invazivne preventivne preglede ter operacije, ki so za določene bolezni v zgodnjem stadiju veliko bolj uspešne kakor v poznejšem.

Učinkovitejša zdravstvena nega – sistem mZdravje lahko na podlagi danih informacij odloča o tem, ali je potreben obisk pri zdravniku ali ne. Na ta način bi lahko preprečili do 30 % nepotrebnih preventivnih zdravniških pregledov. Poleg razbremenjevanja zdravstvenih delavcev s preprečevanjem nepotrebnih pregledov lahko tak sistem sam predlaga rešitve za določene bolj generične težave, ki bi jih zdravnik ob pregledu tudi sam predlagal. K učinkovitosti delovanja zdravstva pa bi pripomoglo tudi posredovanje relevantnih podatkov zdravniku ob zdravstvenem pregledu. Spodbujanje uporabnikov k bolj zdravemu življenju - aplikacija lahko z motivacijskimi motivi in gestami, na primer pohvala ali prikaz napredka, deluje na uporabnikovo psiho, tako da ga stimulira k še večjemu prizadevanju za zdrav način življenja. Sistem lahko uporabniku predlaga in posreduje pomembne informacije glede na njegovo zdravstveno stanje in mu s tem pomaga pri njegovi ozaveščenosti. Težava pri tej točki nastane pri implementaciji takšnega sistema. Sedanji zdravstveni informacijski sistem namreč tega ne predvideva, zato bi bilo treba najprej skrbno načrtovati neki standard v sodelovanju z zdravstveno stroko [14].

2.3 Elektronski zdravstveni karton (EHR)

Elektronski zdravstveni karton je, podobno kot klasični zdravstveni karton, ki ga uporablja naš osebni zdravnik, zbirka pacientovega zdravstvenega stanja, težav in zdravstvene zgodovine. Razlika je v tem, da mora vse specialistične preglede v klasični zdravstveni karton izbrani zdravnik vlagati in zapisovati

sam, v EHR pa lahko vnese izvide in ostale podatke kar specialist po oziroma med pregledom, kar pripomore k bistveno boljši ažurnosti podatkov.

Glavni težavi takšnega sistema pa sta človek in zdravstveni sistem. Težava človeka je, da se mnogokrat zadeve pozabijo zapisati zaradi preobremenjenosti ali stresa, ali pa so zapisani podatki pomanjkljivi oziroma netočni. V zdravstvenem sistemu je v praksi tak informacijski sistem pogosto implementiran nepopolno, kar pomeni, da ga ne podpirajo vsi oddelki ustanove ali posamezni zdravniki v zasebnih ordinacijah. Posamezne implementacije se med seboj lahko razlikujejo po standardih definiranja kliničnih podatkov, zato je to pomembna tema, ki je bolj opisana v poglavju 3. Ker zdravstveni sistem uporabe EHR ne podpira dovolj (ali ga ne zahteva), zdravstvenih informacijskih sistemov mnoge ordinacije ne uporabljajo, kar v praksi pomeni, da se izvidi in ostali dokumenti še vedno prenašajo na papirju.

2.4 Osebni zdravstveni karton (PHR)

Ljudje si za lastne potrebe radi zapisujemo podatke o svojem telesu, saj lahko na ta način s primerjanjem uradnih podatkov o zdravem človeku dobimo sliko, v kakšnem stanju je naše telo. Lahko se odločimo za hujšanje, si zastavimo proces in cilj ter si vsak dan beležimo napredek s tem, ko zapisujemo našo trenutno vrednost telesne mase. Ali pa smo postali ponosni starši in želimo spremljati zdravstveno napredovanje našega otroka, zato si vsak mesec beležimo njegovo višino in težo. Osebe s kroničnimi boleznimi si prav tako zapisujejo nivoje, ki so kritičnega pomena pri določeni bolezni (krvni tlak za srčno-žilne bolezni, krvni sladkor za sladkorno bolezen ...). Teh zapisov pa je lahko veliko, zato nastane težava pri upravljanju s temi podatki, ki jo navadno rešujemo z uporabo računalniških orodij za obdelavo večje količine podatkov, na primer iskanje statističnih pokazateljev ali vizualizacija podatkov. Da pa ne bi potrebovali toliko različnih orodij, kolikor je različnih načinov upravljanja, lahko uporabimo programsko opremo, ki podpira vse operacije skupaj. Takšnemu konceptu pravimo osebni zdravstveni

karton v elektronski obliki.

Kar se tiče medicinskih podatkov, osebni zdravstveni karton vsebuje enake elemente kot EHR. Vsebuje lahko tako uporabnikove lastnosti (na primer višina, teža ali spol) kot bolj specifične, klinične informacije (diagnoze, posege, alergije ...). Razlika je v tem, da osebni zdravstveni karton vodi posameznik sam. Prav zaradi samostojnega vodenja lahko PHR vsebuje informacije, ki niso navedene nikjer drugje, bodisi nepomemben prehlad bodisi alergija na morske sadeže, ki se je prvič pojavila na letošnjem dopustu. Primeren je tako za zdravniški vpogled kakor za osebno vodenje svojega zdravstvenega stanja.

Temeljna razlika med EHR in PHR je, da podatki v sistemih EHR vsebujejo precej več administrativnih podatkov. Ti vsebujejo podatke o pacientovih zavarovanjih, stroških zdravljenj, posegov, medicinsko-tehničnih pripomočkih in podobno. Taki sistemi vsebujejo tudi module za analizo, ki so po eni strani namenjeni pregledu medicinskih podatkov, po drugi strani pa pregledu stroškov poslovanja in pošiljanja različnih statistik državnim institucijam (kot je, na primer Nacionalni inštitut za javno zdravje).

Poglavje 3

Standardi za upravljanje zdravstvenih informacij

Zaradi porazdeljenosti informacij na različnih platformah (konkretno v našem primeru EHR v zdravstveni ustanovi in PHR pri uporabniku) se pojavi potreba po izmenjavi le-teh. Takšna izmenjava elektronskih zdravstvenih kartonov, tako kot vsaka izmenjava katerihkoli podatkov, zahteva poenoten protokol, ki ga poznata tako pošiljatelj kot prejemnik. Zaradi boljše interoperabilnosti, to je izmenjave podatkov med informacijskimi sistemi različnih tehnologij, so bili ustvarjeni standardi, kot so openEHR, HL7, DICOM in mnogi drugi.

3.1 Standard openEHR

Na področju medicinske informatike je standard openEHR odprti standard, ki ima vse specifikacije prosto dostopne, njegova uporaba pa je navadno omejena s pravnimi omejitvami lastnikov standarda, kateri za delo s podatki v sistemih EHR definira večnivojski pristop referenčnega modela z arhetipi in predlogami [25]. Prednost večnivojskega modeliranja je, da so pri procesu ustvarjanja modela prisotni razvijalci in zdravstveni eksperti, vsak na svojem nivoju, razvijalci programske opreme namreč oblikujejo programsko zasnovo,

medtem ko zdravstveni poznavalci oblikujejo semantični del modela. S takim načinom dela se izognemo napakam in nepotrebnemu izgubljanju časa z neznanimi terminologijami ter pomeni, saj vsak sodelujoči opravlja delo svojega področja. Poleg tega pa tudi olajša uporabo zdravstvenim delavcem, saj je vsa terminologija in način dela zasnovan in prilagojen stroki. Referenčni model sam po sebi ne vsebuje nobenih kliničnih informacij, temveč le povezave in splošne atribute. Semantični del sestavljajo arhetipi in predloge, ki jih oblikuje stroka. Posledica tega je generičnost modela in v praksi pomeni, da ga lahko uporabimo za različne domene [24].

Standard openEHR v bistvu le definira način prikazovanja kliničnih informacij in delo z njimi [7], ne definira pa posameznih kliničnih informacij. Kar pomeni, da je treba za uporabo tega standarda s strokovnjaki ustvariti načrt in dokumentacijo vseh posameznih kliničnih konceptov in predstavitvenih modelov v razredih openEHR. To nam omogoča nadaljnjo uporabo podatkov znotraj informacijskih sistemov, ki ta standard uporabljajo, bodisi sistemi EHR bodisi sistemi PHR, ne glede na tehnologijo ali jezik. Gradnja takega modela poteka v jeziku ADL. To je jezik, s katerim na poenoten, standarden način s pomočjo abstraktne sintakse definiramo nove referenčne modele.

Razvoj aplikacij za branje in interpretacijo zdravstvenih kartonov, zapisanih v standardu openEHR, je relativno preprost, saj je treba ustvariti le predstavitev semantičnih modelov (navadno so to preprosti razredi) iz dokumentiranih arhetipov ali predlog ter razviti logiko, ki zna prebrati tak dokument. Slednje je izvedljivo tudi z uporabo raznih prosto dostopnih knjižnic za izbrani programski jezik. Grafični vmesniki zdravstvenih kartonov so prav tako odvisni od dokumentacije modelov, saj je za vnos podatkov treba ustvariti tudi primerne obrazce, kjer moramo vedeti, katere informacije so zahtevane za model oziroma, katere lahko vnašamo programsko. Vsak vnesen podatek moramo shraniti v podatkovno bazo, zato je od podatkovnega modela odvisna tudi implementacija podatkovne baze. Brskanje po podatkovni bazi s kompleksno poizvedbo SQL je potratno, zato raje definiramo semantične

poizvedbe, ki se lahko opravijo kar med branjem in razčlenjevanjem kartona.

3.1.1 Arhetipi

Z arhetipi (ang. `arhetype`) definiramo posamezne klinične informacije ali pa kar celotne skupine kliničnih informacij, ki so strukturirane z minimalnim številom univerzalnih entitet in širokim spektrom možnih podatkovnih entitet za pokritost vseh robnih primerov, kar zagotavlja visoko interoperabilnost in možnost ponovne uporabe arhetipov v različnih kontekstih [23]. Vsak arhetip je lahko definiran z enim od štirih, semantično različnih razredov. Definicija arhetipov je izvedena z jezikom ADL v datotečnem formatu `.adl`.

Kompozicijski razred (ang. `composition class`) služi informacijski strukturi kot vsebnik za arhetipe ostalih razredov. V splošnem ga lahko obravnavamo kot list, ki ga izda zdravnik (na primier izvid, recept ali napotnico). Njegov otrok v strukturnem smislu je sekcijski razred.

Sekcijski razred (ang. `section class`) ima namen opredeljevanja oziroma organiziranja znotraj kompozicijskega razreda. V ta razred zapisujemo arhetipe vnosnih in gručnih razredov.

Vnosni razred (ang. `entry class`) je semantično najbolj pomemben arhetip, saj služi kot vsebnik posameznih informacij. Sestavljen je tako, da je uporaben v različnih sekcijah, ki so med seboj lahko semantično povezane ali ne. Vnosni razred ima med arhetipi najbolj širok pomen, zato je definiran kot abstraktni razred, zato ga je treba razširiti v enega izmed štirih podrazredov.

Opazovanje (ang. `observation`) - najbolj splošen podrazred, navadno ga uporabimo za beleženje simptomov, splošnejših dogodkov, odkritij in meritev. Vsako opazovanje mora vsebovati izmerjeni podatek oziroma glavno informacijo vnosa, stanje subjekta med procesom prejemanja glavne informacije, protokol, ki opisuje proces prejemanja glavne informacije, in časovno informacijo o začetku ter trajanju.

Ocenitev (ang. `evaluation`) - podrazred, ki se uporablja za zapisovanje sklepov in ugotovitev ter interpretacij izvidov. Pogoj za ocenitev je nek izvid oziroma rezultat meritve. Tak arhetip ne vsebuje nobenih informacij o času

ali količinskih vrednostih.

Predpis (ang. `instruction`) - arhetip tega razreda vsebuje informacije o poteku zdravstvenih procesov ali terapij, ki se bodo odvijale (fizioterapija, dieta, predpisana terapija ...).

Ukrep (ang. `action`) - podobno kot predpis ta razred vsebuje informacije o zdravstvenemu procesu ali terapiji, ki je že bila izvedena (na primer aplikacija terapije ali aplikacija cepiva).

Gručni razred (ang. `cluster class`) ima namen enkapsuliranja elementov v semantične gruče, katere se da ponovno uporabiti. Uporabljeni so kot razširitev vnosnih ali drugih gručnih razredov, kjer želimo v en zapis dodati več vsebinsko povezanih informacij. Primer uporabe gručnega razreda je vstavljanje gruče za hranjenje časovnih informacij o začetku, koncu in/ali intervalu k nekemu opazovalnemu podrazredu vnosnega razreda (ta namreč te informacije sam po sebi ne vsebuje, kot je to opisano v razdelku o vnosnih razredih).

Specializacija razreda - včasih lahko naletimo na situacijo, ko potrebujemo neki specifičen arhetip, a so vsi že definirani arhetipi preveč splošni. Da bi se izognili definiranju novega arhetipa, ki bi se od že obstajajočega razlikoval zgolj po enem atributu ali celo samo imenu, lahko uporabimo specializacijo (ang. `specialisation`). Obstoječi arhetip razširimo v nov arhetip, ki podeduje vse atribute starševskega arhetipa ter mu dodamo ali spremenimo atribute, ki so specifični zgolj zanj. Prednost specializacije je tudi iskanje po elementih, saj nam ob iskanju nekega starševskega arhetipa iskanje vrne tudi vse razširjene elemente, a ob iskanju specifičnih razširitev pa ne dobimo korenskih elementov.

3.1.2 Predloge

Nivo nad arhetipi imenujemo predloge (ang. `template`), kateri je v tehničnem pomenu še vedno arhetip, a vsebuje vnaprej določene arhetipe. Tak element je specializiran za svoje področje in je uporabljen kot predloga za zajem točno določenih informacij preko skupka artefaktov. Druga značilnost predloge je

predelava vsebovanih arhetipov, saj lahko posamezne dele odstrani. Ker je predloga v osnovi še vedno arhetip, je njena definicija prav tako izvedena z jezikom ADL.

3.2 IHE

IHE je ameriška neprofitna organizacija, ki si prizadeva za izboljšanje izmenjave podatkov med informacijskimi sistemi oziroma ineroperabilnosti v zdravstvu [8]. Je središče za razprave med razvijalci zdravstvenih informacijskih sistemov, uporabniki in ostalimi zainteresiranimi o izboljšanju načinov integracije heterogenih informacijskih sistemov, ki težijo k temu, da se izboljša nega bolnikov. Iniciativa IHE organizira srečanja (ang. `Connect-a-thon`), kjer imajo razvijalci možnost preverjati povezljivost in komunikacijo med posameznimi sistemi. IHE ima sestavljen okvir, ki opisuje transakcije, ki so obvezne za reševanje realnih medicinskih težav (integracijski profili). Ta okvir ni nič drugega kot dokument z navodili in priporočili, na kakšen način je treba razmišljati in reševati integracijo zdravstvenih informacijskih sistemov z uporabo obstoječih standardov in orodij.

IHE za delo s kliničnimi podatki predlaga uporabo standarda HL7, za prenašanje in ustvarjanje podatkovnih modelov v obliki elektronskih zdravstvenih kartonov, in standard DICOM za kompatibilno prenašanje ter prikazovanje slik.

3.2.1 Standard DICOM

S standardom DICOM je definirano delo s slikami (rentgenske, MR, CT, UZ ...), zajema pa hranjenje (format) in posredovanje (komunikacijski protokoli) slik v elektronski obliki [9]. Razvijalci standarda so se med samim razvojem zgledovali po že uveljavljenem standardu ACR-NEMA (American College of Radiology - National Electrical Manufacturers Association) [21]. Koncepti so bili prilagojeni in izboljšani za modernejšo in bolj generično uporabo.

Standard DICOM za prenašanje informacij v omrežju temelji na uporabi omrežnega protokola TCP/IP, saj je ta protokol najbolj razširjen med uporabniki. Koncept je bil razvit na podlagi standarda ACR-NEMA, ki je temeljil na povezavi le dveh računalnikov (point-to-point). Z razširitvijo na omrežno deljenje podatkov se odpre tudi možnost posredovanja podatkov neposredno pacientom.

Prenašanje slik se lahko opravi tudi brez omrežne povezave z uporabo alternativnih, mrežno neodvisnih medijev, kot so CD-ji. Standard ACR-NEMA takšnega načina prenosa ni predvideval.

Specificirana je tudi interakcija s sistemom preko standardnih ukazov. Koncept standarda ACR-NEMA je predvideval interakcijo le na nivoju samega prenašanja podatkov, standard DICOM pa je vpeljal zahtevo o semantičnih ukazih in pridobivanju točno iskanih informacij. V ta namen je predvidena uporaba servisnih razredov (ang. `service class`).

Standard ACR-NEMA določa minimalno stopnjo skladnosti, to pa je standard DICOM povečal z eksplicitnim opisom strukturiranja oziroma s skladnostnim stavkom (ang. `conformance statement`) za izbiro specifičnih možnosti. Uvedeno je delo z informacijskimi objekti, kar omogoča delo ne samo s slikami, ampak tudi z zvočno-vizualnimi formati, poročili in drugim. Zaradi dela z informacijskimi objekti je bila razvita tehnika za ločevanje med posameznimi tipi, kar omogoča enolično definiranje relacije med objekti.

Standard DICOM je zasnovan modularno po direktivah ISO, zato je nadaljnje razvijanje in razširjanje standarda enostavno in hitro.

3.2.2 Standard HL7

Standard HL7 je podobno kot standard openEHR predviden za interoperabilnost kliničnih informacij. Standard HL7 specificira način, izmenjavo, upravljanje in integracijo zdravstvenih informacij. Njegova uporaba je predvidena predvsem za izmenjavo sporočil in dokumentov med vsemi udeleženci v zdravstvu, ne samo med posameznimi ustanovami, kar omogoča posredovanje in oglaševanje javnih zdravstvenih informacij neposredno vsem sistemom,

ki podpirajo standard HL7. Eden glavnih namenov tega standarda je specifikacija zdravstvenih kartonov v elektronski obliki, ki jih je mogoče prenašati in obdelovati v katerikoli ustanovi s podprtim sistemom.

Standard HL7 vsebuje več vrst standardov, ki so razdeljeni glede na področja, ki jih pokrivajo [21]. Tako poznamo standarde za klinične dokumente (standard HL7 CDA), konceptualne standarde (standard HL7 RIM), standarde za vizualno integracijo aplikacij (standard HL7 CCOW) in standarde za sporočila (standard HL7 verzije 2.x in 3.0). Slednji imajo najpomembnejšo vlogo med vsemi standardi HL7, saj specificirajo način (jezik ter strukturo in tipe podatkov) izmenjave podatkov med sistemi.

Verzija 2.x - bistvo te družine verzij je specifikacija strukture in posredovanja sporočil, namenjenih za podporo administrativnim, logističnim, finančnim in kliničnim procesom, konkretno pa se ga uporablja za komunikacijo med informacijskimi sistemi za administracijo pacientov (PAS), elektronsko upravljanje s praksami (EPM), laboratorijskimi informacijskimi sistemi (LIS), informacijskimi sistemi za obračun, zdravila in diete, farmacevtskimi sistemi ter s sistemi elektronskih zdravstvenih kartonov (EMR) oziroma elektronskih zdravstvenih zapisov (EHR). Vsa dokumentacija je na voljo na uradni spletni strani [22].

Verzija 3.0 - temelj te verzije je formalna metodologija HDF (HL7 Developement Framework) in principi objektno orientiranega razvoja informacijskih sistemov. HDF je okvir, ki definira specifikacije, potrebne za interoperabilnost medicinskih informacijskih sistemov, konkretno dokumentacijo procesov, orodij, pravil in artefaktov, ki se standardno uporabljajo za snovanje ostalih specifikacij HL7.

Pomembna novost je referenčni informacijski model. Podobno kot pri standardu openEHR ima referenčni informacijski model nalogo opredeljevanja podatkov določenih administrativnih in kliničnih procesov z definicijo enolične predstavitve elementov [22].

Spremenjen je bil tudi standard za izmenjavo sporočil, definiran v verziji 2.x. Nova specifikacija zajema osnove in infrastrukture, administrativne ter

klinične domene [22].

HL7 EHR je sistemski funkcijski model, s katerim se definira spisek funkcij, ki so na voljo v sistemu EHR. Predstavlja nabor standardnih opisov posameznih funkcij, ki implementirajo dani sistem EHR, kar omogoča poenostavljen pregled potrebnih lastnosti za komunikacijo z drugim sistemom EHR.

Definicija posameznih semantičnih elementov poteka s standardom HL7 CDA (HL7 Clinical Document Arhitecture), ki je po funkcijski lastnosti podoben arhetipom pri standardu openEHR. S standardom HL7 CDA predstavljamo modele, ki vsebujejo klinične informacije (izvidi, diagnoze, rezultati meritev in podobno), ustvarjene po pravilih, ki jih narekuje referenčni informacijski sistem. Dokumenti so implementirani s formatom XML, ki lahko vsebujejo tudi gradnike HTML za neposredno implementacijo v spletnih aplikacijah. V dokument je omogočeno zapisovanje celotnih dokumentov PDF, ki se lahko uporabijo za nalaganje na naprave, ki ne podpirajo standarda HL7. Slaba stran tega standarda se pokaže pri razumevanju napisanih zdravstvenih kartonov, saj je v primerjavi z definicijo arhetipov in predlog pri standardu openEHR definicija CDA s slabo definiranimi pravili zelo abstraktna in težko nerazumljiva [22].

V tem delu smo za prenos EHR uporabili implementacijo modela HL7 CDA - epSOS. Namen modela je deljenje kliničnih informacij med evropskimi državami. Glavni vsebinski točki implementacije epSOS sta elektronski recepti in skupek pacientovih najpomembnejših zdravstvenih lastnosti.

Pilotni projekt je bil zaključen junija 2014. Na njegovi podlagi je bila izvedena raziskava sistema na področju razumljivosti informacij, uporabnosti in časovnega odziva [13]. Raziskava je pokazala, da se s pomočjo vmesnega medija, ki zna pokazati prevedene zdravstvene informacije, znatno olajša nudenje pomoči tujemu pacientu. Ugotovili so tudi, da je uporabnost takšnega sistema omejena, če je treba za pregled informacij uporabiti ločene aplikacije, zato je priporočena integracija modulov za branje epSOS formata v že obstoječe informacijske sisteme. Predlagana je bila tudi prevajalna funkcija

zdravniških komentarjev in zaznamkov. Ocena zdravniških delavcev je pozitivna, saj menijo, da bi bila implementacija orodja mogoča v vsakodnevno rabo v kliničnem okolju.

Poglavje 4

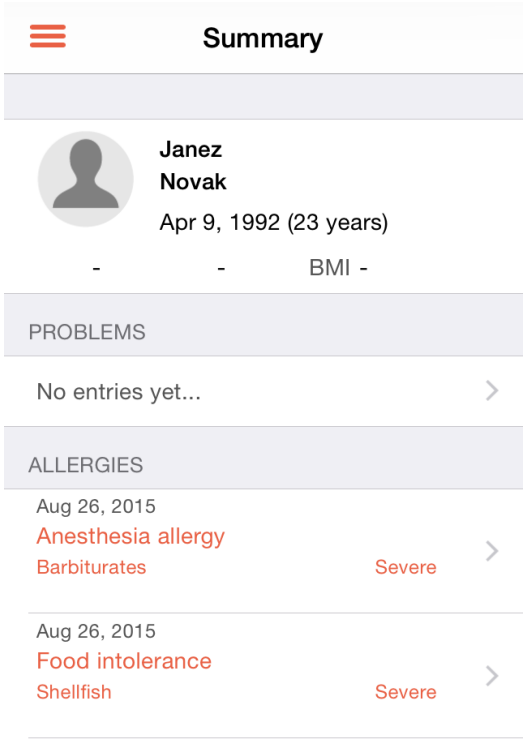
Axilla PHR

V okviru naloge smo izdelali mobilno aplikacijo Axilla. Gre za mobilni osebni zdravstveni karton za platformo Android. Ideja in ime aplikacije nista nova, saj je bila različica za iOS že narejena (Slika 4.1) in je dosegljiva na Applovi spletni trgovini iTunes pod nazivom Axilla PHR – Personal Health Record. Verzija za Android je narejena v skladu s trenutno najnovejšim Googlovim oblikovnim trendom, imenovanim Material Design. Aplikacija je bila v celoti narejena s prostodostopnim orodjem Android Studio. Za pomoč pri izdelavi grafičnega vmesnika so bile uporabljene uradne knjižnice, ki jih zastopa Google, imenovane appcompat podporne knjižnice.

4.1 O tehnologiji Android

Android je operacijski sistem, ki temelji na jedru odprtokodnega operacijskega sistema Linux in je trenutno v lasti podjetja Google. Sistem je namenjen široki paleti mobilnih naprav (na primer pametni mobilni telefoni, tablice ali pametne ure) z zaslonom, občutljivim na dotik. Poleg vseh integriranih funkcionalnosti sistema lahko uporabnik na napravo nalaga programsko opremo oziroma aplikacije (krajše apps) iz spletne trgovine Google Play.

Arhitektura OS je zasnovana kot sklad programskih komponent, ki ga



Slika 4.1: Zaslonska slika aplikacije Axilla na sistemu iOS.

lahko razdelimo na 5 plasti (Slika 4.2):

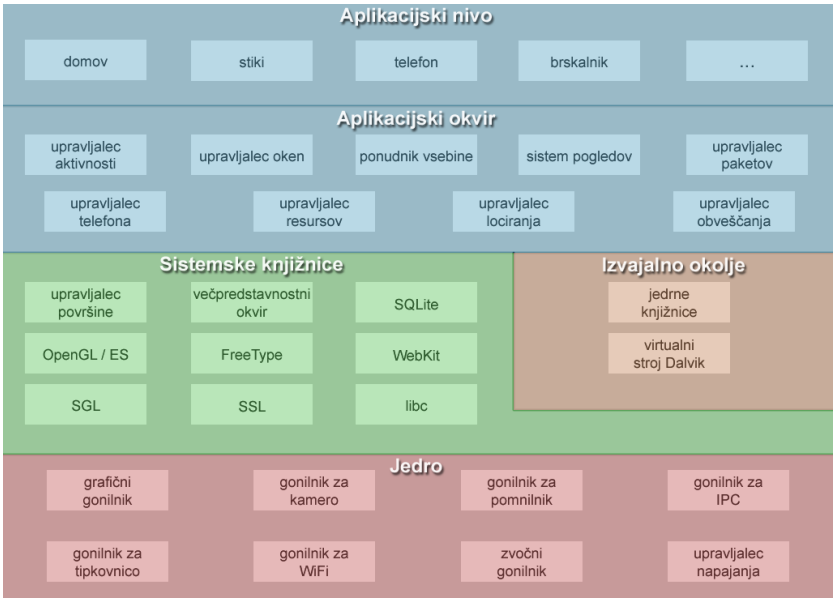
- jedro (ang. `kernel`) Linux z nižje nivojskimi operacijami - ta plast skrbi za uspešno komunikacijo med programsko in strojno opremo (na primer zaslon, kamera, GPS), saj se v tej plasti nahajajo gonilniki naprav;

- sistemske knjižnice - tu najdemo knjižnice, ki aplikacijam nudijo splošna orodja, kot so orodja za dostop do SQLite podatkovne baze ali orodja za implementacijo varnostnih povezav (SSL).;

- izvajalno okolje - v tej plasti se nahajajo knjižnice, specifične za okolje Android. To so knjižnice za razvijanje in poganjanje aplikacij (na primer `android.app`, `android.view`, `android.os`). Poleg knjižnic pa v to plast spada še Dalvik VM, virtualni stroj, ki je zadolžen za poganjanje prevedene javanske kode;

- aplikacijski okvir - ta nudi višje nivojske storitve aplikacijam. Te vključujejo nadziranje poteka aktivnosti, nudenje medaplikacijske komunikacije, dostop do resursov, dostop do sistemskega opozorilnega sistema in nudenje grafičnih sistemskih elementov.

- aplikacije - vrhnjo plast tega modela predstavljajo posamezne aplikacije, nameščene na sistemu;

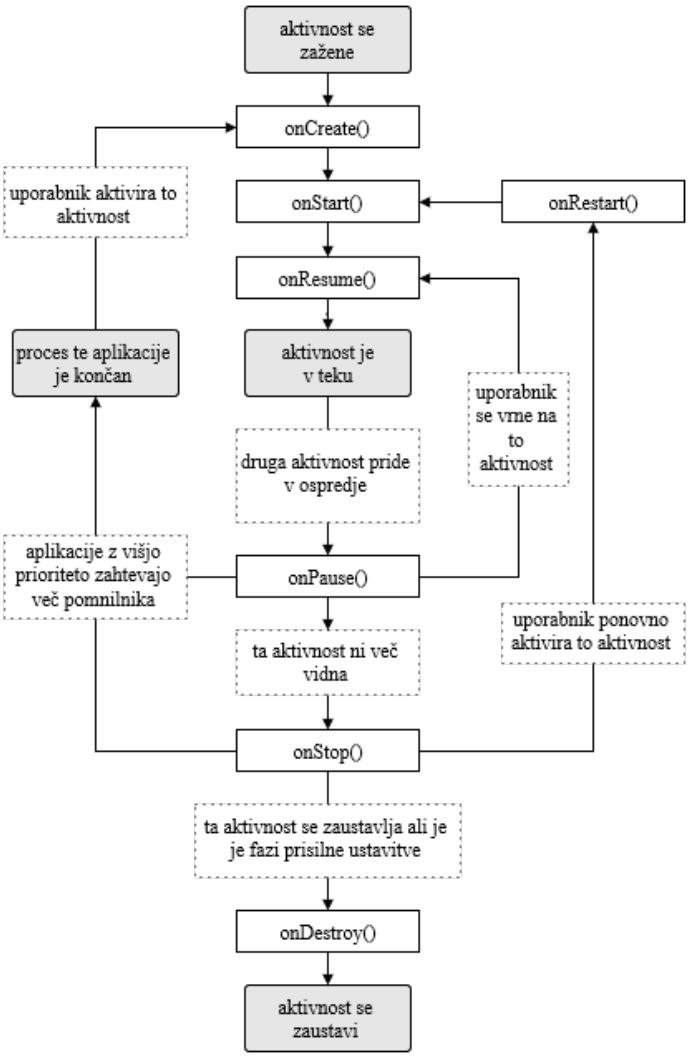


Slika 4.2: Sistemska arhitektura sistema Android, povzeto po [12].

Aplikacija v operacijskem sistemu je lahko napisana v različnih razvojnih orodjih z različnimi programskimi jeziki (C/C++ z NDK, Lua s Corona SDK, HTML, CSS in Javascript s Phonegap, C# z Xamarin ...), v okviru tega dela je bil razvoj aplikacije izveden po predlogih razvijalcev pri Googlu s programskim jezikom Java in orodjem Android SDK v kombinaciji z jezikom XML za izgradnjo grafičnih vmesnikov.

Activity - za izvrševanje aplikacij v sistemu Android je treba pripraviti javanske razrede, imenovane aktivnosti. Ti navadno razširijo v naprej definiran sistemski razred `Activity`. V tem razredu definiramo reference na konkretno datoteko XML, kjer je definirana postavitev grafičnega vmesnika.

Poleg same postavitve lahko definiramo tudi posamezne gradnike preko atributov `id`, kar nam omogoča programsko spreminjanje vmesnika med delovanjem aplikacije, na primer sprememba barve, vsebine, velikosti, dodajanje poslušalcev za dotične geste. Razred `Activity` je zasnovan tako, da se ob ključnih dogodkih kličejo ustrezne metode za obvladovanje teh dogodkov. Ti dogodki so sestavni del življenjskega cikla aktivnosti, predstavljenih v diagramu na Sliki 4.3.



Slika 4.3: Diagram poteka aktivnosti v Androidu, povzeto po [19].

Ob stvaritvi aplikacije se pokliče metoda `onCreate()`, kjer se navadno

postavi grafičen vmesnik in inicializira spremenljivke referenc na gradnike grafičnega vmesnika. Preko parametra metode (razred `Bundle`) lahko v tej fazi obnovimo nastavitve, ki so bile shranjene pred zaprtjem aplikacije. V našem osebnem zdravstvenem kartonu je ta faza večinoma uporabljena za pregledovanje manjkajočih podatkov, ki jih aktivnost potrebuje za prikaz v seznamih. Seznami ostanejo v pomnilniku, zato je dovolj, da do podatkovne baze dostopamo le enkrat, ko je aktivnost ustvarjena.

Sledi klic metode `onStart()`, ko postane aktivnost vidna. Ta metoda je bistvenega pomena za delovanje aplikacije v situaciji, ko je naša aplikacija bila že zagnana, a postavljena v ozadje, bodisi zaradi odprtja nove aktivnosti v odprti aplikaciji bodisi zaradi prekinitve iz druge aplikacije, saj se ob ponovnem zagonu ali priklicu aplikacije v ospredje ne kliče več metode `onCreate()`, ampak zgolj metoda `onStart()`.

Zatem se izvede klic metode `onResume()`, ki zaznamuje priklic aktivnosti v ospredje. V tej fazi je aktivnost v teku in je pripravljena na interakcijo z uporabnikom. Ta faza se izvede tudi po morebitni pavzi, ki se aktivira z raznimi dialogi.

Ko se v ospredje pokliče druga aktivnost ali pa pride do kakšne prekinitve iz druge aplikacije, stanje trenutne aplikacije stopi v pavzo, s tem pa se izvrši metoda `onPause()`. V kolikor je prekinitev bila narejena zaradi dialoga in je v ozadju aplikacije še vedno vidna, se po odstranitvi le-tega ponovno pokliče metoda `onResume()`.

Če pa je bila pognana druga aktivnost v trenutni aplikaciji ali pa povsem nova aplikacija, se trenutna aktivnost prestavi v fazo ustavljanja in pošlje v ozadje ter izvrši metodo `onStop()`. Od tu lahko načeloma pride do dveh situacij. Če ima sistem na voljo dovolj resursov, se aktivnost ohranja v pomnilniku.

Če pride med izvajanjem druge aplikacije do pomanjkanja pomnilniškega prostora, pa lahko za normalno delovanje sistem zaustavi nekaj aktivnosti v ozadju, ki so v stanju čakanja. Takrat potencialna aktivnost pokliče metodo `onDestroy()`. To metodo lahko, zaradi sproščanja pomnilnika ali pa za-

radi preprostega končanja aktivnosti, razvijalec pokliče tudi sam s sistemsko metodo `finish()`.

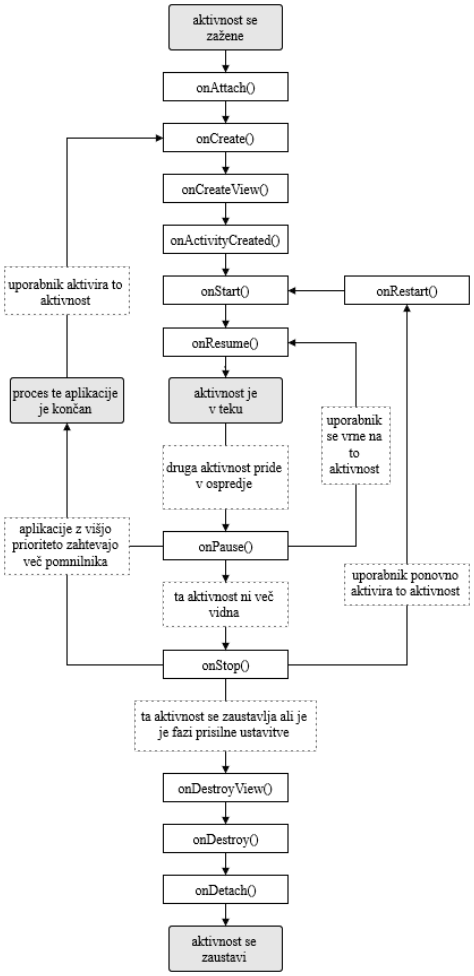
Fragment - s prihodom sistema verzije 3.0 in nivoja API 11 se je pojavil tudi koncept fragmentov. To so javanski razredi, ki sami po sebi ne morejo obstajati, ampak potrebujejo neko aktivnost, ki lahko fragment vsebuje, njihov glavni namen pa je povečati fleksibilnost uporabniških vmesnikov. API 11 je bil primarno razvit za tablične računalnike, saj se je zaradi večjega zaslona pojavila dilema, ali povečati prikaz vsebine ali enostavno prikazati več vsebinskih elementov naenkrat. Slednje je bila glavna motivacija za razvoj fragmentov. Tako je omogočeno na napravah z manj prostora enostavno menjati vsebinske elemente med seboj v enem vsebniku, na večjih napravah pa lahko ustvarimo več vsebnikov in vsak prikazuje svojo vsebino. Tako kot aktivnost lahko tudi fragment vsebuje svoj grafični vmesnik formata XML, ni pa to nujno.

Fragment je po delovanju podoben aktivnosti, saj se prav tako na sistemske dogodke odziva z dogodkovnimi metodami `onCreate()`, `onStart()`, `onResume()`, `onPause()`, `onStop()` in `onDestroy()`, dodatno pa vsebuje še metode, ki skrbijo za dinamično ustvarjanje fragmenta [11], kar prikazuje diagram na Sliki 4.4.

Prva razlika je vidna že ob začetku izvajanja fragmenta, saj se pred klicem metode `onCreate()` izvrši metoda `onAttach()`, ki prejme in shrani referenco na svojo starševsko aktivnost. V tej fazi je smiselno preverjati, ali ima starš implementirane vmesnike, ki jih potrebuje fragment za medobjektno komunikacijo, ali ne in primerno reagirati.

Po klicu `onCreate()` se izvede metoda `onCreateView()`, ki skrbi za izris grafičnih elementov. Inicializacija grafičnih gradnikov je navadno izvedena v tej metodi, saj v metodi `onCreate()` nimamo dostopa do postavitve vmesnika, `onCreateView()` pa kot parameter prejme LayoutInflater, s katerim po razširitvi dobimo dostop do postavitve in do njenih elementov.

Sledi `onActivityCreated()`, ki v praksi pogosto dopolnjuje pomen metode `onCreateView()`, kjer je treba počakati, da se postavi specifičen set



Slika 4.4: Diagram poteka fragmenta v Androidu, povzeto po [11].

gradnikov, ki si med pripadajočimi elementi deli podatke. Nadaljevanje poteka je podobno navadni aktivnosti vse do faze ustavljanja, kjer se po metodi `onStop()` proži metoda `onDestroyView()`, ki skrbi za ustrezno odstranitev fragmenta iz grafičnega vmesnika. Po izvedbi te metode je za prikaz novega fragmenta ponovno treba izvršiti metodo `onCreateView()`.

Na koncu po metodi `onDestroy()` se izvede še klic na `onDetach()`, ki poskrbi za ustrezno odstranitev referenc iz starševske aktivnosti.

Manifest - ob prvem zagonu aplikacije je treba zagnati prvo aktivnost aplikacije. Tej aktivnosti rečemo zagonska aktivnost, ki jo definiramo

v manifestu aplikacije, ki se nahaja v datoteki formata XML, imenovani `AndroidManifest.xml`. Definicija je izvedena preko značk `<activity>` z uporabo sistemske konstante `LAUNCHER` (odsek kode 4.1).

```
<activity android:name=".activities.MainActivity">
  <intent-filter>
    <category android:name="android.intent.category.LAUNCHER" />
```

Odsek kode 4.1: Definicija zaganjalne aktivnosti.

Poleg definiranja zagonske aktivnosti nam ta datoteka hrani tudi spisek referenc na vse ostale aktivnosti, do katerih lahko med izvajanjem aplikacije uporabnik dostopa, katere vsebujejo referenco starševske aktivnosti, definirano z atributom `parentActivityName` v znački `<activity>`.

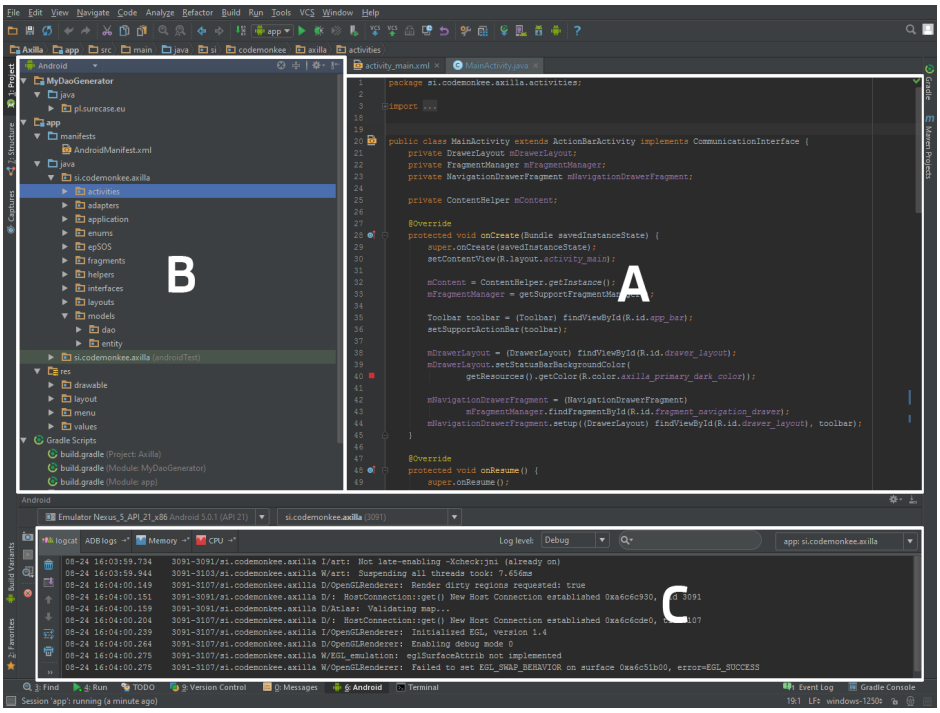
Mnoge aplikacije, vključno z Axillo, potrebujejo za svoje delovanje omrežno povezavo, a za dostop do storitev naprave je treba aplikaciji odobriti ustrezne pravice. Odobritev prav tako definiramo v manifestu z uporabo značk `<uses-permission>` in izberemo želeno pravico iz nabora pravic, ki jih omogoča model android.permission. Manifest vsebuje tudi podatke o stopnjah API, označene z značko `<uses-sdk>` z atributi `minSdkVersion` in `targetSdkVersion`.

4.2 Android Studio

Za razvoj aplikacije smo uporabili IDE Android Studio. Razvojno orodje Android Studio je uradno razvojno okolje za razvoj aplikacij v sistemu Android [3], ki temelji na razvojnem okolju IntelliJ IDEA proizvajalca JetBrains. Omogoča hitro pisanje kode s samodejnim dopolnjevanjem spremenljivk, metod ali razredov. Nudi samodejno vstavljanje blokov kode, ki jo je treba implementirati ali prepisati, saj nam med samim kodiranjem razvojno orodje samodejno hrani reference do vseh že definiranih razredov. Podpira tudi generiranje novih datotek na preprost način z vmesnikom, kjer se izbere ime

in tip razreda, ki ga želimo ustvariti, vključno z možnostjo razširjanja že generiranih razredov. Nov projekt lahko v Android Studiu generiramo za različne platforme, ki podpirajo sistem Android, in sicer za mobilne telefone in tablice, televizorje, pametne ure ter očala (Google Glass).

Najbolj pomembni deli vmesnika za razvoj so prikazani na Sliki 4.5.



Slika 4.5: Razčlemba grafičnega vmesnika Android Studio.

Prostor, označen s črko A, je prostor, v katerem razvijalec prebije večino časa. To je prostor, kamor se vpisuje koda (Java, XML) ali se oblikuje grafični vmesnik aplikacije s postavljanjem grafičnih elementov.

Prostor B služi kot raziskovalec datotek in si ga lahko prilagajamo na različne načine predstavitve projekta. Na voljo je:

- standardni raziskovalec, kjer je struktura datotek predstavljena kot običajno logično drevo na disku,
- način pogleda projekta, kjer je prikaz prilagojen strukturi javanskih paketov in

- način Android, kjer je drevesna struktura projekta prikazana glede na posamezne aplikacije in skripte. Ta prikaz je za razvoj aplikacij Android najbolj praktičen.

C označuje prostor, kjer se nahajajo konzole za sledenje aplikacijam. Tu so prikazane informacije glede Gradle procesov, izpisi iz naprave, na kateri se poganja aplikacija v razhroščevalnem načinu. Ta prostor hrani tudi orodje za verzioniranje kode (Subversion ali Git).

Za izgradnjo projekta (ang. `project build`) je uporabljeno orodje Gradle. Glavna naloga tega orodja je povezovanje in grajenje izvršilnih aplikacij APK iz datotek v projektu, med razvijanjem pa skrbi tudi za ažurirano gradnjo zunanjih knjižnic, z ukazom dependencies (odsek kode 4.2). Primer uporabe zunanjih knjižnic za aplikacijo Axilla:

```
dependencies {
  compile fileTree(dir: 'libs', include: ['*.jar'])
  compile 'com.android.support:appcompat-v7:21.0.+'
  compile 'com.android.support:recyclerview-v7:22.+'
  compile 'com.android.support:cardview-v7:22.+'
  compile 'de.greenrobot:greendao:1.3.7'
}
```

Odsek kode 4.2: Integracija podpornih knjižnic.

Vsak posamezen projekt vsebuje datoteko `build.gradle`, kjer so definirana navodila za grajenje aplikacije, napisana pa so v označevalnem jeziku Groovy.

4.3 Združljivost aplikacije

Pri izdelavi aplikacije za sistem Android je za definiranje združljivosti treba določiti najmanjšo možno stopnjo API (ang. `API level`). Stopnja API določa programsko ogrodje, ki jo ponuja različica sistema. Pod programsko

ogrodje spadajo javanski razredi in paketi, XML elementi ter pravice, ki jih aplikacija potrebuje za dostopanje do perifernih naprav ali storitev.

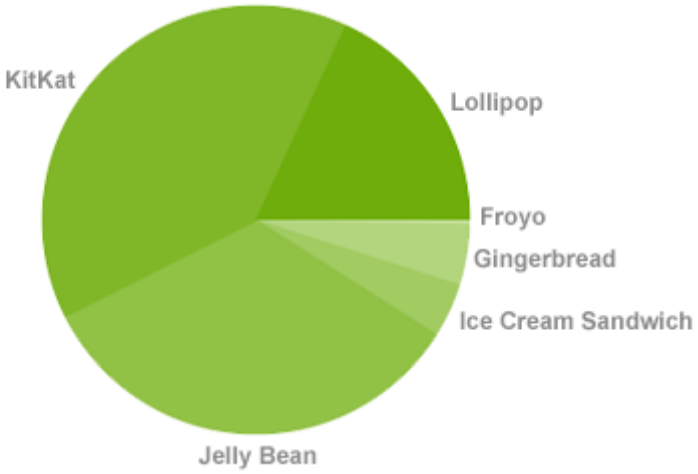
Podjetje Google pogosto opravlja raziskavo o uporabi naprav s sistemom Android [2] in po zadnjih raziskavah v času, ko je ta naloga pisana, kot je razvidno iz Grafa 4.6 in Tabele 4.1, je približno tretjina trga še vedno na sistemu Jelly Bean, dobra tretjina uporablja različico KitKat in le 15,5 % najnovejši Lollipop. Iz tega razloga smo aplikaciji Axilla za najnižjo stopnjo izbrali stopnjo 16, da bi pokrili čim večje število uporabnikov, a pri tem ne bi izgubili veliko sistemskih orodij višjih stopenj API.

Različica	Naziv	`API`	Delež
2.2	Froyo	8	0,3 %
2.3.3 - 2.3.7	Gingerbread	10	4,6 %
4.0.3 - 4.0.4	Ice Cream Sandwich	1	4,1 %
4.1.x		16	13,0 %
4.2.x	Jelly Bean	17	15,9 %
4.3		18	4,7 %
4.4	KitKat	19	39,3 %
5.0	Lollipop	21	15,5 %
5.1		22	2,6 %

Tabela 4.1: Rezultati raziskave o uporabi OS Android dne 3. avgust 2015 [2].

Ker je bila aplikacija razvita v skladu z oblikovnimi smernicami Material Design, kjer je veliko gradnikov in funkcionalnosti odvisnih od sistemskih razredov in metod, ki so implementirane v razvojni stopnji API 21, smo bili primorani uporabiti podporne knjižnice (ang. `support library`) [4], ki jih nudi podjetje Google ravno v namene združljivosti s starejšimi sistemi.

Materal Design je skupek smernic za snovanje grafičnih vmesnikov operacijskega sistema Android, ki za prikaz gradnikov uporablja abstraktno predstavitev papirja, kar da občutek globine in dejanske interakcije z materialom



Slika 4.6: Graf deležev uporabe Android operacijskih sistemov, povzeto po [2].

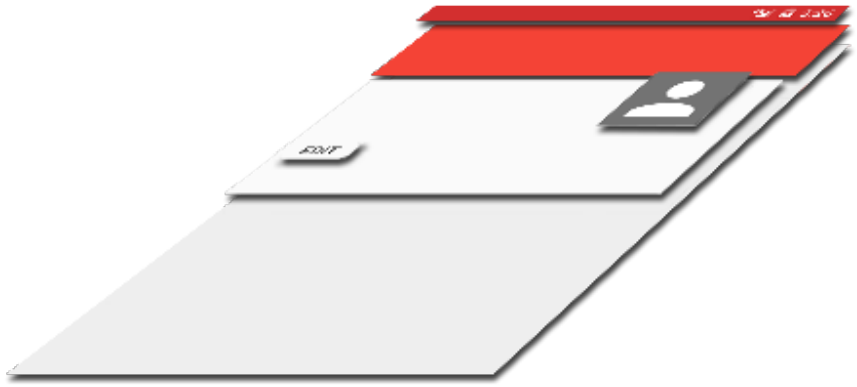
[6] (podobno kakor interakcija s knjigo), namesto interakcije, ki temelji na abstraktnih elementih, ki lebdijo v nekem praznem prostoru. Slika 4.7 prikazuje razširjen pogled posameznih delov primera aplikacije, narejene po principih Material Design.

Privzeto je uporabljen pri sistemskem vmesniku sistema verzije 5.0 oziroma Lollipop, kar pa ne pomeni, da je omejen samo na to platformo. Material Design je namenjen uporabi na vseh platformah, mobilnih telefonih, tabličnih računalnikih, stacionarnih računalnikih ali preko spleta.

4.4 Razvoj podatkovnega modela

Za hranjenje podatkov je uporabljena podatkovna baza SQLite, ki jo sistem podpira in je že integrirana v sistemskih paketih. Podatkovni model je razdeljen na tri dele, kjer vsak del prispeva k drugačni funkciji aplikacije. Prvi del zajema relacije med entiteto uporabnika (predstavljen s tabelo User) in zdravstvenimi vnosi:

- alergije, v tabeli `Allergy`,
- diagnoze, v tabeli `Diagnosis`,



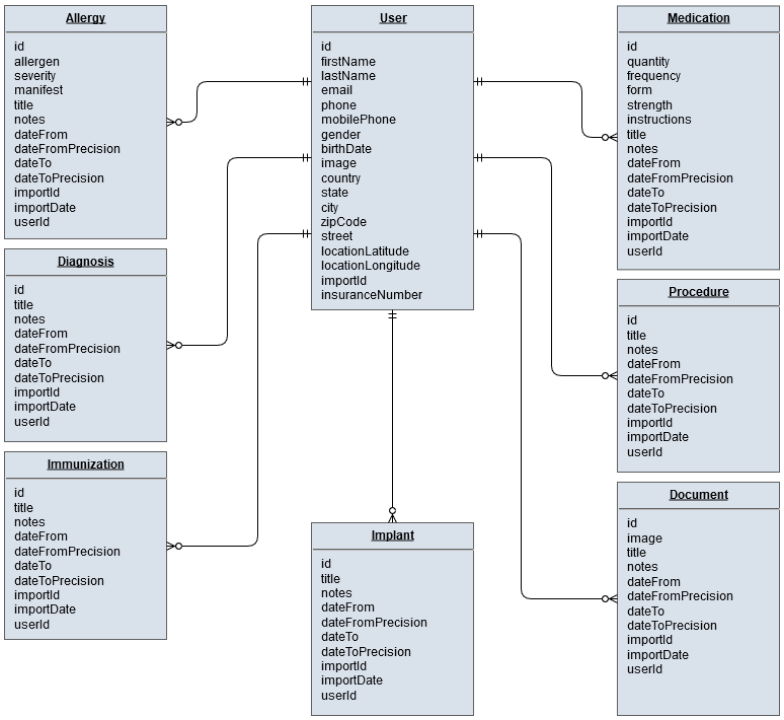
Slika 4.7: Shema plasti gradnikov Material Design, povzeto po [10].

- imunizacije (vakcinacije), v tabeli `Immunization`,
- vsadki, v tabeli `Implant`,
- zdravila, v tabeli `Medication`,
- posegi, v tabeli `Procedure` ter
- dokumenti in zapiski, v tabeli `Document`.

Modeli so razviti z uporabo abstraktnega modela `MedicalEntry`, ki vsebuje lastnosti, ki so skupni vsem zgoraj naštetim zdravstvenim vnosom, zajemajo pa atribute `id`, `title`, `notes`, `dateFrom`, `dateTo`, `dateFromPrecision`, `dateToPrecision`, `importId`, `importDate` in `userId`, kjer `id` predstavlja primarni ključ tabele, `userId` pa tuji ključ, ki referencira uporabnika. Za entitete, ki predstavljajo diagnoze, imunizacije, vsadke in posege je dovolj le uporaba predloge atributov v razredu `MedicalEntry`, ostale entitete pa je treba razširiti z atributi, ki so specifični za domeno, ki jo entiteta predstavlja. Entiteto alergij je potrebno razširiti z atributi alergena, resnosti alergije in manifestacijo alergijske reakcije. V tabeli za zdravila so dodani atributi za količino in frekvenco jemanja terapije, obliko in moč zdravil ter navodila

za jemanje. Entiteta dokumentov pa poleg atributov razreda `MedicalEntry` vsebuje še atribut za slike.

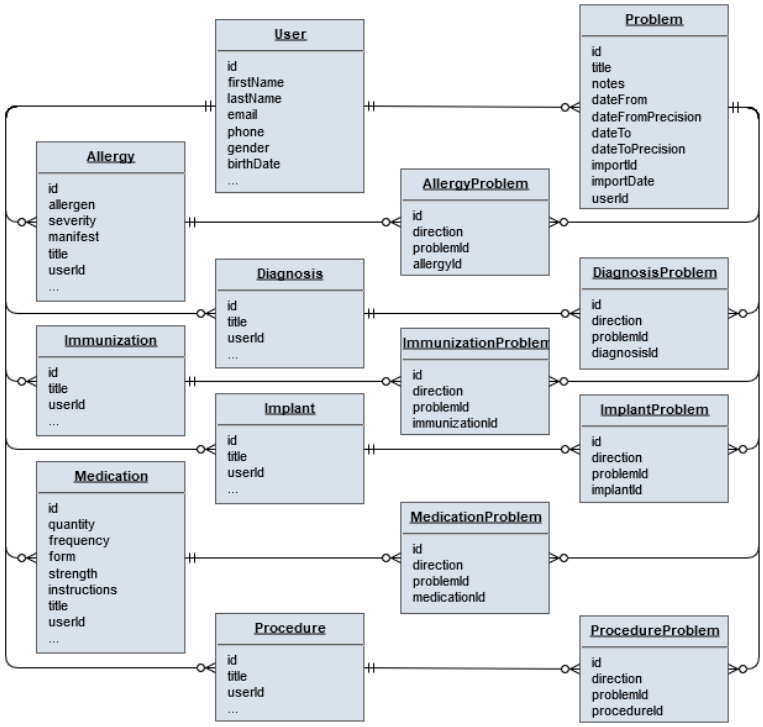
Hrbtenica aplikacije je tabela, ki hrani vse uporabnike. To je tabela `User`, ki hrani osnovne podatke o uporabniku (ime, priimek, e-poštni naslov, telefonsko številko, spol, rojstni datum, sliko) in njegove geografske podatke (država, zvezna država, mesto, poštna številka, ulica ter zemljepisna širina in višina). V namene uvažanja podatkov iz sistemov EHR se hranijo tudi podatki o uvoznem ključu in številki zdravstvenega zavarovanja. Relacija med uporabnikom in kliničnimi podatki je predstavljena na Sliki 4.8. Tak način implementacije relacij omogoča preprosto iskanje po podatkih določene vrste kliničnih vnosov, ki jih uporabnik zahteva za prikaz v aplikaciji, glede na trenutno izbranega uporabnika.



Slika 4.8: ER diagram uporabnikov in kliničnih vnosov.

Drugi del podatkovnega modela predstavlja relacijo med uporabnikovimi težavami in posameznimi kliničnimi vnosi. Uporabnik lahko naniza poljubno

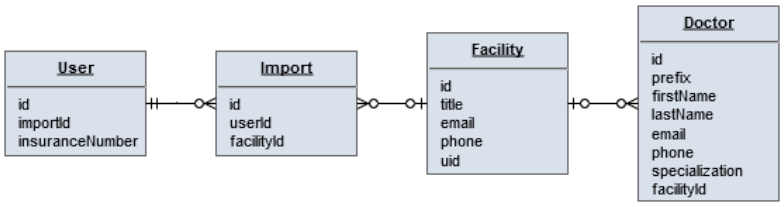
število težav in jih asociira z, na primer diagnozo ali alergijo. To je realizirano z relacijo mnogo proti mnogo, kjer se za implementacijo uvede vmesna tabela, ki hrani povezave na obe tabeli, ki sta med seboj povezani. Takšna topologija omogoča iskanje po vseh pacientovih težavah, če jih želimo le videti ali če želimo posamezne težave povezati z določenim kliničnim vnosom, hkrati pa lahko pri pregledovanju enega vnosa obenem pridobimo vse težave, ki so povezane z njim. Diagram ER, ki predstavlja ta del podatkovnega modela, je prikazan na Sliki 4.9.



Slika 4.9: Diagram ER uporabnikov in težav.

Zadnji pomemben del podatkovnega modela pa skrbi za uvažanje podatkov iz storitve epSOS. Predstavljata ga relaciji 1:M med tabelama uporabnikov in uvozov ter 1:M med tabelama zdravstvenih ustanov in uvozov. Uvozi (tabela Import) so predstavljeni z enolično oznako `id`, identifikacijsko številko uporabnika in identifikacijsko številko ustanove. Implementacija predstavlja relacijo M:N med tabelama uporabnikov in ustanovami. Poizvedovanje se

tako lahko razširi tudi na iskanje zdravnikov, saj je vzpostavljena relacija 1:M med ustanovo in zdravniki. Slika 4.10 predstavlja implementacijo podatkovnega modela, ki skrbi za uvažanje e-kartonov iz zunanjih virov.



Slika 4.10: Diagram ER uporabnikov, uvozov ter ustanov in pripadajočih zdravnikov.

Za razvoj ogrodja za delo s podatkovno bazo oziroma ogrodja ORM smo uporabili prosto dostopno knjižnico GreenDao [5]. To je javanska knjižnica, ki je namenjena generiranju modelov ORM za sistem Android in podatkovno bazo SQLite. Za izdelavo entitet je treba najprej ustvariti shemo z razredom `Schema` in ji definirati izhodno pot (javanski paket). Na koncu je potrebno še zagnati generacijo ogrodja z metodo `generateAll()`, ki prejme kreirano shemo in dodaten argument. Proces predstavlja odsek kode 4.3.

```
Schema schema = new Schema(1, ENTITY_PACKET_STRING);
schema.setDefaultJavaPackageDao(DAO_PACKET_STRING);
// Generacija entitet...
new DaoGenerator().generateAll(schema, args[0]);
```

Odsek kode 4.3: Osnovna konfiguracija za kreacijo PB.

Posamezno entiteto se ustvari z razredom `Entity`, ki jo je treba dodati prej ustvarjeni shemi z metodo `addEntity()`. Entiteti je treba postaviti še primarni ključ, to pa se izvede z metodo `addIdProperty()`. Za definicijo posameznih atributov so pripravljene metode, katere obenem specificirajo tudi podatkovni tip atributa (niz, celo število, datum, decimalno število in ostali podprti tipi v SQLite). Relacije med entitetami se ustvarijo z metodami

`addToMany()` in `addToOne()`, ki prejmeta za parametra drugo entiteto in atribut, ki predstavlja identifikacijsko številko, uporabljeno za tuji ključ. V odseku kode 4.4 je primer kode za generiranje entitete, ki opisuje diagnozo.

```
Entity diagnosis = schema.addEntity("Diagnosis");
Property id = diagnosis.addIdProperty().getProperty();
diagnosis.addStringProperty("title");
diagnosis.addStringProperty("notes");
diagnosis.addDateProperty("dateFrom");
diagnosis.addIntProperty("dateFromPrecision");
diagnosis.addDateProperty("dateTo");
diagnosis.addIntProperty("dateToPrecision");
diagnosis.addStringProperty("importId");
diagnosis.addDateProperty("importDate");
Property userId = diagnosis
    .addLongProperty("userId")
    .getProperty();
user.addToMany(diagnosis, userId);
problemDiagnosis.addToOne(diagnosis, Id);
Property problemDiagnosisId = diagnosis
    .addLongProperty("problemDiagnosisId")
    .getProperty();
diagnosis.addToMany(problemDiagnosis, problemDiagnosisId);
```

Odsek kode 4.4: Primer definicije entitete (`Diagnosis`) z orodjem GreenDao.

Generator ustvari javanske razrede, ki predstavljajo entitete v podatkovni bazi, s konstruktorji in ostalimi metodami za pridobivanje razrednih podatkov. Ustvarjeni so tudi razredi DAO, ki vsebujejo metode za izvrševanje operacij CRUD nad podatkovno bazo, z uporabo jezika SQL ter metode za vzdrževanje relacij med entitetami. Generator poskrbi tudi za razreda `DaoMaster` in `DaoSession`, s katerima se v aplikacij ustvari bazo in drži sejo za interakcijo. Slednja omogoča predpomnenje za hitrejše dostopanje do že

prebranih podatkov. Primer generacije baze v aplikaciji je prikazan v odseku kode 4.5. Knjižnica je dosegljiva na repozitorijih GitHub na naslovu `https://github.com/greenrobot/greenDAO`.

```
Context context = getApplicationContext();
DaoMaster.DevOpenHelper helper =
    new DaoMaster.DevOpenHelper(context, DB_NAME, null);
SQLiteDatabase sqLiteDatabase = helper.getWritableDatabase();
DaoMaster daoMaster = new DaoMaster(sqLiteDatabase);
```

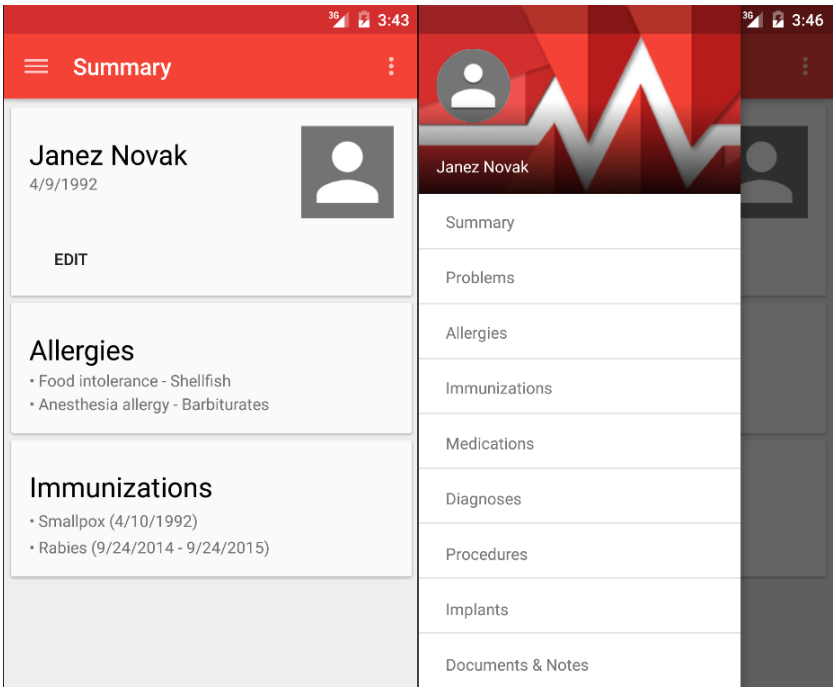
Odsek kode 4.5: Generacija nove podatkovne baze v aplikaciji.

4.5 Razvoj in delovanje aplikacije

Delovanje aplikacije se zgleduje po uporabi zdravniškega kartona. To pomeni, da morajo biti zdravniku ali zdravstvenemu delavcu pomembni zapisi izpostavljeni, da se omogoči hitro ukrepanje brez odvečnega iskanja po celem dokumentu. Najbolj pomembni so povzetki o alergijah in kroničnih težavah. Navedba informacij o pacientovih razvadah (na primer pitje ali kajenje) tudi lahko zdravniku pomaga pri izbiranju anestetika. Lahko bi vsebovala tudi podatke o krvni skupini in vrednosti meritev, na primer meritev krvnega sladkorja ali tlaka, pri čemer se pokaže najmanjšo vrednost meritev, ki niso starejše od enega meseca. Tudi vakcinacije, ki jih še ni treba obnoviti, so lahko prikazane na povzetku. Prikazane pa bi bile lahko tudi resnejše težave v smislu zapletov [18].

Ker so priporočila eno in realizacija projekta drugo, je bilo treba narediti nekaj kompromisov. Aplikacija zato ob zagonu, ko se uporabniku najprej prikaže stran s povzetki, prikazuje le osnovne informacije uporabnika, spisek alergij in še vedno aktivne imunizacije (Slika 4.11).

Po priporočilih oblikovnega koncepta Material Design se na levi strani zaslona nahaja navigacijski meni, od koder lahko uporabnik zamenja vsebino



Slika 4.11: Zaslonski sliki strani o povzetku in navigacijskem meniju.

glavnega okna. Na voljo so povezave do povzetkov, težav, alergij, imunizacij, zdravil, diagnoz, posegov in dokumentov (Slika 4.11).

Aplikacija je sestavljena tako, da je večino časa aktivna le ena aktivnost (glavna aktivnost), ki vsebuje dva ločena vsebnika, enega za navigacijski meni in enega za prikaz vsebine. Implementacija tega sistema je izvedena z uporabo fragmentov, saj sta tako pogled vsebine kot navigacijski meni fragmenta. Za zamenjavo vsebin sta ključna dva postopka. Prvi je identifikacija pritisnjene povezave in aktivacija grafičnega vmesnika, drugi pa je generacija ustreznega fragmenta. Med generiranjem je treba priskrbeti tudi podatke za prikaz, do katerih se dostopa s posameznimi razredi DAO. Ob pritisku na element v navigacijskem meniju se pokliče metoda `navigate()` vmesnika (ang. `interface`) `CommunicationInterface` (odsek kode 4.6). Ta vmesnik implementira glavna aktivnost in definira metodo `navigate()` (odsek kode 4.7), kjer se izvede tudi posodobitev grafičnega vmesnika s klicem metode `closeDrawers()`.

```
interface CommunicationInterface {
  void navigate(Content contentIndex);
  void refresh();
}
```

Odsek kode 4.6: Vmesnik `CommunicationInterface`.

```
void navigate(Content content) {
  // Nastavi globalno lastnost trenutne vsebine.
  mContent.setCurrentContent(content);
  // Generira nov fragment na podlagi trenutne vsebine.
  Fragment contentFragment = mContent.createFragment();
  if (contentFragment != null) {
    // Zapre navigacijski meni.
    mDrawerLayout.closeDrawers();
    // Zacne izvajati proces zamenjave fragmenta.
    int containerViewId = R.id.fragment_content_container;
    int drawerStringsId = R.array.drawer_item_strings;
    String tag = getContentTag(content);
    FragmentTransaction transaction =
        mFragmentManager.beginTransaction();
    transaction.replace(containerViewId, contentFragment, tag);
    transaction.commit();
  }
}
```

Odsek kode 4.7: Implementacija metode `navigate()`.

Fragment, zadolžen za prikazovanje podatkov kliničnih vnosov, je definiran z razredom `ContentDisplayFragment`. Ob vsaki zamenjavi vsebinskega fragmenta se ustvari nova instanca razreda, ki ji je treba definirati kontekst

(trenutno izbrana klinična vsebina) z uporabo javanskih razredov enum (odsek kode 4.8).

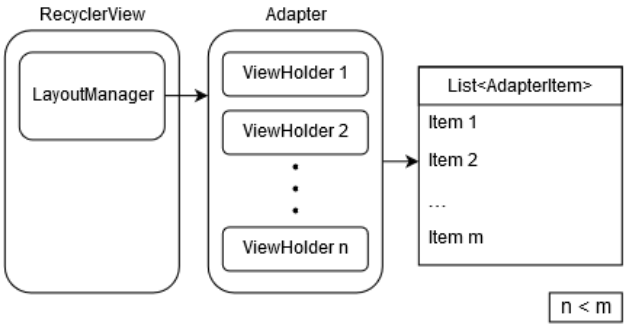
```
enum Content {
  SUMMARY, PROBLEMS, ALLERGIES, IMMUNIZATIONS, MEDICATIONS,
  DIAGNOSES, PROCEDURES, IMPLANTS, VITALS, DOCS_AND_NOTES
}
```

Odsek kode 4.8: Enum, ki predstavlja konstante vsebine.

Aktivnosti, ki predstavljajo obrazce za vnos novih kliničnih vnosov, razširjajo abstraktni razred `AbstractInputActivity`. Ta definira osnovno obnašanje aktivnosti, kot je vračanje na starševsko aktivnost, inicializacija skupnih gradnikov, menjava načina izbranega vnosa (načina za pregledovanje in urejanje) in definicija metod za upravljanje z uporabnikovimi težavami, ki so vezane na izbran kliničen vnos, vsaki aktivnosti, ki ta razred razširja, pa narekuje tudi implementacijo metode `saveToDatabase()`, ki skrbi za shranjevanje vnosa v podatkovno bazo.

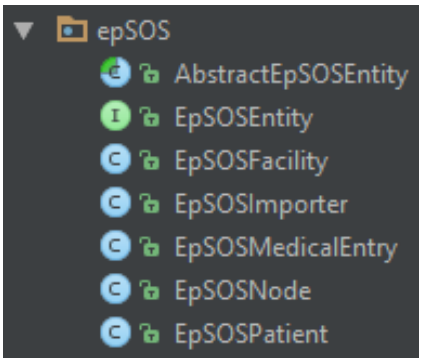
Vsi seznami v aplikaciji implementirajo razred `RecyclerView` (abstrakcija implementacije tega razreda prikazuje Shema 4.12), ki podatke pridobi preko razreda `Adapter`, ta pa vsebuje grafične elemente `ViewHolder` in seznam podatkov, ki jih je treba prikazati. Podatke predstavlja preprost javanski objekt, ki enkapsulira skupek spremenljivk, potrebnih za prikaz enega vnosa. Število elementov `ViewHolder` je fiksno in je odvisno od prostora na grafičnem vmesniku, saj se med premikanjem po seznamu menjajo le objekti, ki vsebujejo podatke, grafičnim elementom pa se prestavljajo le postavitveni indeksi.

Uvažanje podatkov poteka preko storitve REST, kjer se iz strežnika prenese epSOS datoteka, ki vsebuje zdravstveni karton. Tega aplikacija prebere in obdela s pomočjo modula epSOS (struktura paketa je prikazana na Sliki 4.13).



Slika 4.12: Shema splošne implementacije seznamov, povzeto po [1].

Glavni razred predstavlja EpSOSImporter, ki implementira metode za branje, tolmačenje in shranjevanje podatkov ter koriščenje storitev REST v metodi `getXmlStream()` (odsek kode 4.9). Ta iz naslova URL, številke zdravstvenega zavarovanja in pin številke, ki jo pacient dobi pri zdravniku, sestavi naslov URI, ki ga nastavi odjemalcu HTTP kot ciljno pot, na katero se pošlje zahtevek za pridobitev datoteke.



Slika 4.13: Vsebina paketa za uvoz epSOS datotek.

```
InputStream getXmlStream(String url,
                         String kzz, String pin) {
  AndroidHttpClient androidHttpClient =
      AndroidHttpClient.newInstance(AGENT);
  Uri uri = buildUri(url, kzz, pin);
  HttpHost host =
      new HttpHost(uri.getHost(), SSL_PORT, uri.getScheme());
  HttpGet request = new HttpGet(uri.toString());
  try {
    HttpResponse response =
        androidHttpClient.execute(host, request);
    HttpEntity entity = response.getEntity();
    if (entity != null) {
      return new ByteArrayInputStream(
          EntityUtils.toByteArray(entity));
    } else {
      return null;
    }
  } catch (IOException e) {
    return null;
  } finally {
    androidHttpClient.close();
  }
}
```

Odsek kode 4.9: Postopek za koriščenje storitve REST.

Branje datoteke XML poteka rekurzivno s sprotnim preverjanjem poti `xPath`, ki je definirana v vsakem posameznem objektu (`EpSOSFacility`, `EpSOSMedicalEntity` in `EpSOSPatient`), z metodo `findNodesRecursive()`, ki je prikazana v odseku kode 4.10. Ta vrne seznam objektov `EpSOSNode`, ki predstavljajo izbrano pot, izluščeno iz drevesne strukture, ki jo predstavlja epSOS datoteka.

```
List<EpSOSNode> findNodes(EpSOSNode node,
                          String[] xPath, int i) {
  List<EpSOSNode> nodes = new ArrayList<>();
  if (index < xPath.length) {
    for (EpSOSNode childNode : node.getChildNodes()) {
      if (childNode.equals(xPath[i])) {
        nodes.addAll(findNodes(childNode, xPath, i + 1));
      }
    }
  } else {
    nodes.add(node);
  }

  return nodes;
}
```

Odsek kode 4.10: Rekurzivna metoda za branje datoteke epSOS.

Entitetni razredi `EpSOSFacility`, `EpSOSMedicalEntity` in `EpSOSPatient` razširjajo abstraktni razred `AbstractEpSOSEntity`, ki vsebuje splošne operacije za delo z entiteto. Ob kreiranju objekta se iz seznama EpSOSNode objektov preberejo in zabeležijo posamezni podatki, nato pa se objekt shrani v podatkovno bazo.

Poglavje 5

Zaključek

V diplomskem delu smo raziskali in predstavili stanje na področju mobilnih medicinskih aplikacij po svetu. Uporaba IT rešitev na širokem področju zdravstva in načina življenja je večinoma še vedno v zgodnjih fazah hitro rastočega razvoja. Ugotovili smo, da je lahko razvoj aplikacij mZdravja danes izjemno donosen pod pogojem ustrezno izbrane ciljne skupine uporabnikov, poleg tega pa je iz tehnološkega vidika vsak dan več možnosti za integracijo našega življenja s pametnimi napravami in s tem pomeni tudi izboljševanje življenjskega standarda.

V sklopu diplomskega dela je bil uspešno zasnovan in razvit prototip mobilne aplikacije osebnega zdravstvenega kartona Axilla za operacijski sistem Android, ki predstavlja ekvivalent že obstoječi aplikaciji, razviti za sistem iOS. V ospredju je vizualna interpretacija grafičnega vmesnika in način delovanja, ki se med sistemoma precej razlikujeta, saj le-ta omogoča in poenostavlja rabo enakih funkcionalnosti na način, ki je uporabniku priljubljenega sistema predvsem domač. Z uporabo podpornih knjižnic, ki so za sistem primerno prirejene in ne zasedajo ogromno prostora v pomnilniku, je stabilno izvajanje aplikacije zagotovljeno za večino uporabnikov sistema Android, zagotovljen pa je tudi enak videz in občutek.

Pomemben aspekt mobilnega zdravstvenega kartona je prenašanje informacij med sistemi. Za prototip smo uspešno razvili in vanj implementirali

modul za interpretacijo podatkov o pacientu po evropskem standardu za obdelavo kliničnih podatkov epSOS, za katerega se je pokazalo, da ima na področju Evropske unije visoko verjetnost širše uporabe. Razviti prototip je temelj za nadaljnji razvoj aplikacije do produkcijske faze primarno za Evropsko unijo.

Aplikacija v trenutni fazi nudi možnost vstavljanja in pregledovanja uporabniškega zdravstvenega stanja, konkretno alergij, imunizacij, zdravil, diagnoz, posegov in vsadkov, kar v praksi pomeni, da je prototip pripravljen za uporabo na območju EU, saj popolnoma podpira podatkovni koncept modela epSOS. Prototip bi bilo smiselno nadgraditi še z možnostjo hranjenja večjih dokumentov in slik, življenjskih znakov (na primer srčni utrip ali krvni sladkor) ter telesnih lastnosti (denimo telesna teža in višina). Naslednji korak pri razvoju aplikacije pa je razvoj strežniške podpore, ki nudi hranjenje kliničnih podatkov v oblaku. To bi znatno izboljšalo uporabniško izkušnjo, saj bi se lokalne slike in datoteke, ki mobilni napravi z omejenim pomnilnikom porabijo veliko prostora, po sinhronizaciji s strežnikom pomanjšale ali stisnile, da bi ostali le povzetki dokumentov ali sličice nižje resolucije.

Ker prototip upošteva standard epSOS, je z aplikacijo, ki je razvita za sistem iOS, povsem kompatibilen s stališča prenašanja kliničnih podatkov ali celotnega uporabniškega računa. To pomeni, da lahko uporabnik, ki je vodil svoj zdravstveni karton na sistemu iOS, svoj račun izvozi v datoteko XML, jo posreduje novi napravi s sistemom Android in jo uvozi v svojo aplikacijo.

Literatura

[1] Android: Creating lists and cards. https://developer.android.com/training/material/lists-cards.html. [Spletna objava, dostopano: avgust 2015].

[2] Android dashboards. https://developer.android.com/about/dashboards/index.html. [Spletna objava, dostopano: avgust 2015].

[3] Android studio. https://developer.android.com/sdk/index.html. [Spletna objava, dostopano: december 2014].

[4] Android support libraries. https://developer.android.com/tools/support-library/features.html. [Spletna objava, dostopano: januar 2015].

[5] greendao – android orm for sqlite. http://greendao-orm.com/. [Spletna objava, dostopano: marec 2015].

[6] Material design. https://www.google.com/design/spec/material-design. [Spletna objava, dostopano: avgust 2015].

[7] openehr. http://www.openehr.org. [Spletna objava, dostopano: avgust 2015].

[8] Organizacija ihe. http://www.ihe.net/. [Spletna objava, dostopano: avgust 2015].

[9] Standard dicom. http://dicom.nema.org/. [Spletna objava, dostopano: avgust 2015].

[10] This is material design. http://googledevelopers.blogspot.com/2014/06/this-is-material-design.html. [Spletna objava, dostopano: avgust 2015].

[11] Murat Aydin. *Android 4: New features for Application Development.* Packt Publishing Ltd, 2012.

[12] Stefan Brahler. Analysis of the android architecture. *Karlsruhe institute for technology*, 2010.

[13] European Commission. Digital agenda for europe: Cross-border health project epsos: What has it achieved? http://ec.europa.eu/digital-agenda/en/news/cross-border-health-project-epsos-what-has-it-achieved. [Spletna objava, dostopano: avgust 2015].

[14] European Commission. Green paper on mobile health ("mhealth"). http://ec.europa.eu/digital-agenda/en/news/green-paper-mobile-health-mhealth, April 2014. [Spletna objava, dostopano: avgust 2015].

[15] European Commission. What mhealth can do for you. http://europa.eu/rapid/press-release_MEMO-14-266_en.htm, April 2014. [Spletna objava, dostopano: avgust 2015].

[16] European Commission. Eu countries' mhealth app market ranking 2015. http://mhealtheconomics.com/eu-countries-mhealth-app-market-ranking-2015/, May 2015. [Spletna objava, dostopano: avgust 2015].

[17] Vincenzo Della Mea. What is e-health (2): the death of telemedicine? *Journal of Medical Internet Research*, 3(2), 2001.

[18] dr. Majda Ambrož Mihelčič. osebna komunikacija. [Zdravstveni dom dr. Julija Polca Kamnik, 23. 4. 2015.

[19] Christian Fritz, Steven Arzt, Siegfried Rasthofer, Eric Bodden, Alexandre Bartel, Jacques Klein, Yves le Traon, Damien Octeau, and Patrick McDaniel. Highly precise taint analysis for android applications. 2013.

[20] Panagiotis Germanakos, Constantinos Mourlas, and George Samaras. A mobile agent approach for ubiquitous and personalized ehealth information systems.

[21] I. Indrajit and B. Verma. DICOM, HL7 and IHE: A basic primer on Healthcare Standards for Radiologists. *Indian Journal of Radiology and Imaging*, 17(2):66–68, 2007.

[22] Health Level Seven® INTERNATIONAL. Hl7 standards. `http://www.hl7.org/implement/standards/product_matrix.cfm`. [Spletna objava, dostopano: avgust 2015].

[23] Catalina Martínez-Costa, Marcos Menárguez-Tortosa, and Jesualdo Tomás Fernández-Breis. An approach for the semantic interoperability of {ISO} {EN} 13606 and openehr archetypes. *Journal of Biomedical Informatics*, 43(5):736 – 746, 2010.

[24] delovna skupina Odbora za zdravstveno-informacijske standarde Ministrstvo za zdravje. Primerjava standardov hl7 : openehr in priporočila za uveljavljanje standardov v zdravstveni informatiki v sloveniji. `https://unimed.mf.uni-lj.si/ozis/system/files/HL7-OpenEHR%20v4.1%2020091130_final.pdf`, November 2009. [Spletna objava, dostopano: avgust 2015].

[25] J.D. Trigo, C.D. Kohl, A. Eguzkiza, M. Martinez-Espronceda, A. Alesanco, L. Serrano, J. Garcia, and P. Knaup. On the seamless, harmonized use of iso/ieee11073 and openehr. *Biomedical and Health Informatics, IEEE Journal of*, 18(3):872–884, May 2014.